COMMUNICATIONS

SIGGRAPH '92 SHOWCASE

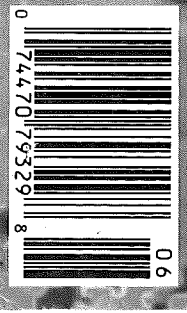
Parallel Database Systems

Developments on the Intellectual Property Front

Collective Dynabases

Fast File Processing

U.S. $6.95
Can. $8.50



June 1992
VOLUME 35, NUMBER 6

OF THE ACM

Stimulate your imagination at the world's premier conference on computer graphics and interactive techniques. Delight your senses in Chicago– a city of diverse and captivating images. Experience it all at SIGGRAPH '92, July 26-31.

Explore the latest and greatest in computer graphics. Challenge your mind in courses. Experience new, unpublished research presentations. Participate in candid discussions and debates. Evaluate leading-edge hardware, software, and systems. Enjoy the diversity of technologically advanced art.

Go beyond the limits with guerrilla technologies. Examine high-speed, networked problem-solving. Look into the future at a learning lab for kids. Enjoy the best of the best in international theater performance, film, and video.

There is something of interest for everyone at SIGGRAPH '92.

- Visualization, Modeling, and Simulation
- Visual Communication and Multimedia
- Visual Computer Systems and Networks
- Visualization Policies, Ethics, and Standards

City of Chicago 3D graphic data courtesy of Skidmore, Owings & Merrill; Rendering by T. Clark and C. Brown, Midwest Litho Arts; *Galaxy Map* M. J. Geller, J. P. Huchra, V. de Lapparent, R. K. McMahan, and E. E. Falco, Harvard Smithsonian Center for Astrophysics

INSIGHT THROUGH IMAGES

Wind Trajectory Ribbons B. Hibbard and B. Paul, Space Science and Engineering Center, University of Wisconsin at Madison; *Fire and Clouds* G. Y. Gardner, Grumman Data Systems; *Waves* C. Wedge and C. Ludwig, Blue Sky Productions, Inc.

Please send conference information to: (Members of ACM SIGGRAPH will automatically receive conference materials in April and need NOT return this coupon.)

Name (please print) _____ Organization _____

Address _____ City _____

State/Province _____ Country _____ Postal Code _____

Phone _____ Fax _____ Email _____

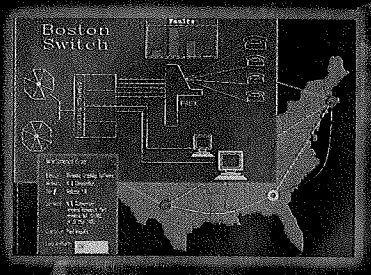
Clip coupon and mail to SIGGRAPH '92, Conference Management, 401 North Michigan Avenue, Chicago, IL 60611 USA, phone: (312) 321-6830, fax: (312) 321-6876, email: info92@siggraph.org CACM

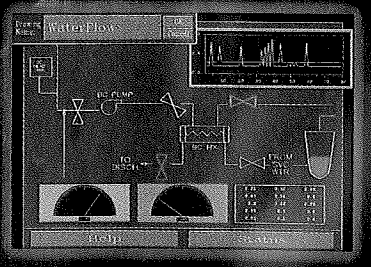
92 ACM SIGGRAPH '92
Conference
July 26-31
1992
Exhibition
July 28-30
1992

Sponsored by the Association for Computing Machinery's Special Interest Group on Computer Graphics in cooperation with the IEEE Computer Society's Technical Committee on Computer Graphics

IS IT A GUI, OR IS IT GODZILLA?

Why spend thousands of hours developing your own GUI only to find you've created a monster. Instead, spend your time, money and effort wisely and build your reputation with the market leader. V.I. Corporation, the pioneer in dynamic graphics interface development environments, developed DataViews to cut your graphical interface development time by up to 80% while producing a dynamic interface that is easy to use, extend or modify. DataViews is the only interface development solution for monitoring real-time processes that offers:





Versatility And Portability With A Comprehensive Tool Set - DataViews comprises DV-Draw, the user friendly interactive prototyper and drawing editor, and DV-Tools, an extensive library of 'C' routines. With DV-Draw you can build displays without programming, creating your own look-and-feel with input objects like menus and sliders. You can include Motif or Open Look widgets and import CAD drawings and images. With DV-Draw's rapid prototyping capabilities you can continuously refine your user interface and save valuable time and effort. With DV-Tools you can manage your DV-Draw screens and build your run-time application. You can run your Dataviews applications in virtually any UNIX or VMS environment.

Real Time Data Output - DataViews displays dynamic data output by using graphs and meters; by changing the scale or color of objects, or by the movement or swapping of objects, to name a few. DV-Draw contains more than 60 pre-defined graph types suitable for displaying x-y output, analog output, and time series output from your application.

Extensibility - DataViews is a proven GUI development environment with more than 7000 users. DataViews lets you change or enhance your interface application and lets your end-users modify the screens that you developed.

Support, Consulting And Training - We provide comprehensive technical support, training and consulting services. And we have more than 30 locations worldwide making it convenient for you to do business with us.

DataViews
V.I. Corporation

Call Us NOW! 1-413-586-4144 for DataViews

WE'RE MONITORING THE WORLD

In Europe Call 44-734-892111

JUNE 1992
COMMUNICATIONS
OF THE ACM

CONTENTS

SPECIAL SECTION
SIGGRAPH '92 Showcase

ARTICLES

COLUMNS

DEPARTMENTS

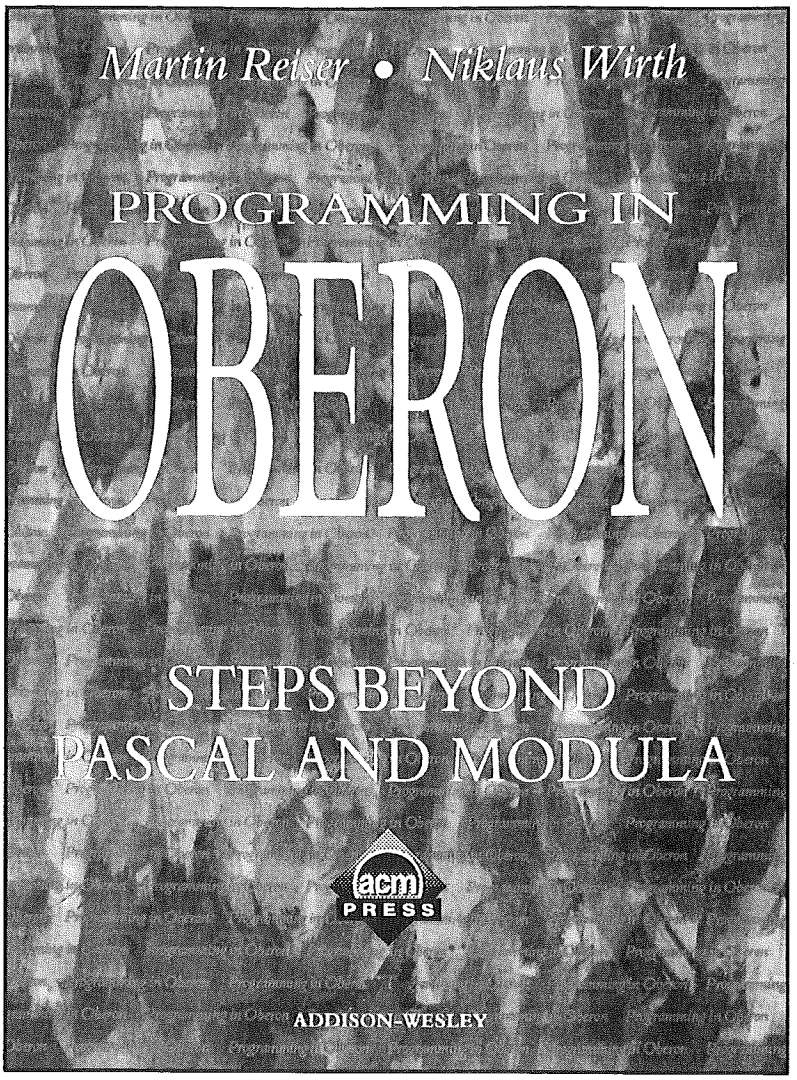
Introducing... the second volume in the Oberon Trilogy!

Programming in Oberon

Steps Beyond Pascal and Modula

Martin Reiser, IBM and ETH, Zurich, and Niklaus Wirth, ETH

This definitive guide and tutorial to programming in the Oberon language—designed as the successor to Pascal and Modula-2—is co-authored by Martin Reiser and Niklaus Wirth, the latter being the creator of these languages. Through a tutorial approach, the authors focus on the unique features of the Oberon language—the concept of type extension and its support of an object-oriented style of programming—while illustrating the continuity with Pascal and Modula.



ACM Press Books Order Form

ACM Press
1515 Broadway, New York, NY 10036-9998

Please send me ___ copy(ies) of Programming in Oberon – Steps Beyond Pascal and Modula (ACM Order # 706921) at $37.75 for nonmembers, $33.95 for ACM Members. Shipping waived when prepaid by check.

Ship to:

Name _____

Company _____

Address _____

City/State _____ Zip _____

☐ My check or money order for the total order, $_____, is enclosed. Because I've prepaid, shipping is free.

☐ Charge to the account listed below. I understand that I'll be charged for shipping.

☐ VISA ☐ MasterCard ☐ American Express

Acct. # _____ Expires _____

Signature _____

ACM Member No. _____

Prices subject to change, and good in the U.S. only.

CACM 6/92

COMMUNICATIONS OF THE ACM
A monthly publication of the Association for Computing Machinery

Publications Office
ACM, 1515 Broadway,
New York, New York 10036
(212) 869-7440 FAX: (212) 869-0481
email jimm@acmvm.bitnet

Director: Mark Mandelbaum
Associate Director: Janet Benton

Executive Editor: James Maurer
Senior Editor: Diane Crawford
Associate Editors: Linda Feczko, Thomas E. Lambert
Copy Editor: Carolyn Lieberman
Editorial Assistant: Robert Fox
Calendar Items: Mickey Abbate

Senior Writer: Karen A. Frenkel
Art Director: Dennis Ahlgrim
Production Manager: Lynn D'Addesio
Business Manager: Roxanne Carcaterra
Copyright: Jane Carlton
Advertising Manager: Walter Andrzejewski

Advertising Display Representatives:
New England and Mid-Atlantic:
Cascio Communications
(312) 332-5945; Fax: (312) 332-6009
Contact Dean Horowitz
Midwest and South:
Cascio Communications
(312) 332-5945; Fax: (312) 332-6009
Contact Lori McGinnis
West:
Marshall Rubin & Associates
(818) 782-1511; Fax: (818) 782-2827
Contact Marshall Rubin
Recruitment Display Advertising
Community Advertising—Arthur Rosen
(201) 487-2511; Fax: (201) 487-0099
Advertising Associates—Philip Friedman
(201) 461-5422; Fax: (201) 461-5886
All Foreign
ACM
(212) 869-7440; Fax: (212) 302-5826
Classified Advertising: ACM Advertising Department, Donald Werkheiser or Carla Roberts, 1515 Broadway, New York, NY 10036. (212) 869-7440; Fax: (212) 869-0481.
Communications of the ACM (ISSN 0001-0782) is published monthly by the Association for Computing Machinery, Inc., 1515 Broadway, New York, NY 10036. Second class postage paid at New York, NY 10001, and other mailing offices. Postmaster: Please send address changes to Communications of the ACM, 1515 Broadway, New York, NY 10036.
Subscriptions: Annual subscription cost of $30.00 is included in the society member dues of $75 (for students, cost is included in $22.00 dues); the nonmember annual subscription is $109. See top line of mailing label for subscription expiration date coded in four digits: the first two are year, last two, month of expiration. Microfilm editions through 1980 can be purchased from Microfilm Dept., Publishing Services Div., Waverly Press, 428 East Preston Street, Baltimore, MD 21202. Microfilm and microfiche are also available from University Microfilms International, 300 North Zeeb Road, Dept. PR, Ann Arbor, MI 48106; (800) 521-0600.
Single copies are $8 to members and $17 to nonmembers. Please send orders prepaid plus $4 for shipping and handling to ACM Order Dept., P.O. Box 64145, Baltimore, MD 21264. For credit card orders call (800) 342-6626 (in Baltimore, Alaska, or Canada, call (301) 528-4261). Order personnel on duty 8:15–4:45 EST. After hours, please leave message and order personnel will return your call.

BPA
acm PRESS

EDITORIAL POINTERS

June *Communications* features four articles on Showcase, a research exhibition of multiuser, interactive, networked, visual computational science projects, and a part of the ACM SIGGRAPH '92 Annual International Conference on Computer Graphics and Interactive Techniques.* This month's special section, "Visualization in Networked Environments," organized by Guest Editor John C. Hart of the Electronic Visualization Laboratory and National Center for Supercomputing Applications, covers the four distinct Showcase themes—local heterogeneous networked environments ("Metacomputing"), remote visualization and collaboration paradigms ("The Distributed Laboratory"), novel visual interfaces ("The Cave"), and specific applications ("Modeling and Analysis of Empirical Data in Collaborative Environments"). For those who will not be attending SIGGRAPH '92, this is an excellent representation of the research that makes up the exhibits, for those who will be there, have fun!

Many readers responded to the The League for Programming Freedom's January 1992 "Viewpoint: Against Software Patents" in "ACM Forum." Additional pros and cons in the intellectual property and software patent controversies are addressed by professor of law Pamela Samuelson and coauthors in "Legally Speaking: Developments on the Intellectual Property Front." A position paper "Debunking the Software Patent Myths" by Paul Heckel, author of *The Elements of Friendly Software Design* (Sybex Inc. 1991), while motivated by the League's January "Viewpoint" voices an overall defense of software patents.

Three distinctive articles in the broad area of databases round out the issue. "An Object-Oriented Methodology for Knowledge Base/Database Coupling" by Kunihiko Higa and coauthors investigates the difficult and critical task of achieving a natural and effective knowledge base/database coupling using an object-oriented strategy. Next, authors DeWitt and Gray discuss the viability of highly parallel database machines as well as some unsolved research issues in "Parallel Database Systems." Finally, David Ferguson in "Bit-Tree: A Data Structure for Fast File Processing" discusses one of the variants of the B-Tree, the standard organization for indexes in a database system.

NEXT MONTH: July's *Communications* focuses on the controversial encryption standard proposed by the National Institute of Standards and Technology. The issue will outline the proposed Digital Signature Standard (DDS) and feature relevant commentaries from people in the encryption field.

James Maurer

Executive Editor

*SIGGRAPH '92 will take place July 26 to July 31, 1992 at Chicago's McCormick Place Convention Center. For information on how to register or get an advance program for SIGGRAPH '92 see the advertisement in this issue, call the SIGGRAPH information hotline at (312) 321-6830, or email: info92@siggraph.org.

ADVANCING HUMAN CAPABILITIES THROUGH INFORMATION TECHNOLOGY

Founded in 1947 as the society of the computing community, the Association for Computing Machinery is dedicated to the development of information processing as a discipline, and to the responsible use of computers in an increasing diversity of applications. The purposes of ACM are to advance the sciences and arts of information processing, to promote the free interchange of information among specialists and the public, and to develop and maintain the integrity and competence of individuals in the field.

COMMUNICATIONS OF THE ACM

A monthly publication of the Association for Computing Machinery

Submission Information: All manuscripts should be submitted to the Executive Editor following the Information for Authors printed in the January 1992 issue.

D792

MEMBERSHIP APPLICATION

ADVANCING HUMAN CAPABILITIES **acm** *THROUGH INFORMATION TECHNOLOGY*

Mail to:
ACM, P.O. Box 12114
Church Street Station
New York, NY 10257

You may use this ACM Membership Application to: 1) Join ACM and take advantage of the special Publication and SIG rates for ACM members; 2) Join the Special Interest Group(s) of your choice at the SIG Member (non-ACM member) rates on the reverse side; individuals joining SIGs using these rates are not entitled to ACM membership privileges.

Please print

First, Middle Initial, Last Name

Address

Address

City / State /Zip or City / Country / Postal Code

( ) Day Telephone Number

For Office Use Only

ACM Membership

ACM Membership includes a subscription to the monthly *Communications of the ACM*

☐ **$77.00 Voting Member Applicants**
You must subscribe to the Purposes of ACM; have attained professional stature as demonstrated by intellectual competence and ethical conduct in the arts and sciences of information processing; and must satisfy at least one of the requirements at the right.

1. Bachelor's Degree. Institution:
2. Equivalent level of education. Institution:
3. Four full-time years of experience (attach a brief statement).

I attest the above is correct
Signature

☐ **$77.00 Associate Member Applicants**
You must subscribe to the purposes of ACM. Associate Members may convert to Voting Member status at any time by writing ACM Headquarters for a "Self-Certification" form.

☐ **$23.00 Student Member Applicants**
You must be registered in an accredited educational institution and a Faculty Member must certify your full-time status.

Institution:
Faculty Member's Signature
Expected Graduation Date: ____/____ (MM YY)

Check member class desired

☐ **$72.00 Joint Membership—IEEE-CS**
Members of the IEEE-CS receive a $5 dues discount. (not Affiliates with "N9" member #'s).

Society:
Member #:

☐ **$62.00 Joint Membership—International Computing Societies**
ACS (Australia), ADIG (Guatemala), AFCET (France), AICA (Italy), API (Portugal), ATI (Spain), BCS (United Kingdom), BIRA/IBRA (Belgium), CIPS (Canada), CSIS (Czechoslovakia), CSZ (Zimbabwe), GI (Germany), HKCS (Hong Kong), ICS (Ireland), IPA (Israel), IPSJ (Japan), NGI (Netherlands), NZCS (New Zealand), SCCC (Chile), SCS (Shanghai).

☐ **Spouse Membership:** For more information, please write to ACM Membership Services Department, 1515 Broadway, New York, NY 10036

SIG Membership

Non-ACM Members Only

☐ **SIG Membership only. (Non-ACM Members only.)**
Membership in special interest group(s) of your choice includes a newsletter subscription. SIG non-ACM members are not eligible for ACM SIG member rates or ACM membership privileges. *See "SIG Member (non-ACM) Rates" on reverse.*

Mailing List

Optional

ACM occasionally makes its membership list available to companies and societies for computer-related mailings. If you wish to restrict the use of your name for these purposes, please check one of the following:

☐ ACM announcements only.
☐ ACM and sister society announcements only.

For Office Use: Recruit/Promotion Code
CACM-6/92

Purposes of ACM

Signature Required

Note: Membership dues include $32.00 ($23.00 for students) toward a subscription to *Communications of the ACM.*

To advance the sciences and arts of information processing; to promote the free interchange of information about the sciences and arts of information processing both among specialists and among the public; and to develop and maintain the integrity and competence of individuals engaged in the practice of information processing.

I hereby affirm that I subscribe to the purposes of ACM and understand that my membership is not transferable.

Signature *Date*

Effective 7/92

(OVER)

Publications

Air Options: Available to overseas members. Circle desired rate(s) and add to publication subscription and/or SIG membership rate(s).

Partial Air Services - Air freight to Amsterdam and Dutch surface mail. Available only to Europe, India, Africa, and Mideast.

Full Air Service - Air service from U.S. Available to all overseas locations including Hawaii.

Circle appropriate rate(s) and indicate subtotal. Membership includes a subscription to ACM's flagship publication Communications of the ACM. If air service is desired for member copy of CACM, circle appropriate Air Option.

Add Appropriate Overseas Air Option(s) to Voting/Associate or Student Rate(s)

	Code #	Voting/ Associate	Student	Partial	Full Air	Total
Communications of the ACM (monthly) addt'l subscriptions only	101	$26.00	$21.00	+ $30.00	$60.00	= $
Membership includes a subscription to *CACM*						
Journal of the ACM (quarterly)	102	23.00	18.00	+ 20.00	25.00	= $
Computing Surveys (quarterly)	103	16.00	11.00	+ 15.00	30.00	= $
Computing Reviews (monthly)	104	35.00	30.00	+ 20.00	50.00	= $
Collected Algorithms, Vols I, II, III, IV, V & 1 yr's quarterly updating supplements	105	220.00	195.00	+ 10.00	20.00	= $
ACM Guide to Computing Literature (annual) You will receive the current published edition.	160	98.00	93.00	no air offered		= $
ACM No-Nonsense Guide to Computing Careers	120	20.00	15.00	+ 6.00	8.00	= $
Transactions on: (all quarterlies)						
Mathematical Software /TOMS	108	27.00	22.00	+ 10.00	20.00	= $
Database Systems /TODS	109	26.00	21.00	+ 10.00	20.00	= $
Programming Languages and Systems /TOPLAS	110	25.00	20.00	+ 10.00	20.00	= $
Graphics /TOG	112	31.00	26.00	+ 10.00	20.00	= $
Information Systems /TOIS	113	27.00	22.00	+ 10.00	20.00	= $
Computer Systems /TOCS	114	27.00	22.00	+ 10.00	20.00	= $
Software Engineering & Methodology /TOSEM	115	23.00	18.00	+ 10.00	20.00	= $
Modeling & Computer Simulation /TOMACS	116	31.00	26.00	+ 10.00	20.00	= $
Letters on Programming Languages and Systems / LOPLAS*	117	26.00	21.00	+ 10.00	20.00	= $

**Expected Publication Date May, 1992*

PUBLICATION SUBTOTAL $ _____

Special Interest Groups (SIGs)

Note: SIG members who are not ACM members do not receive a subscription to *Communications of the ACM.*

Definitions:
Air Options: Available to overseas members. Circle desired rate(s) and add to publication subscription and/or SIG membership rate(s).

Partial Air Service - Air freight to Amsterdam and Dutch surface mail. Available only to Europe, India, Africa, and Mideast.

Full Air Service - Air service from U.S. Available to all overseas locations including Hawaii.

Circle appropriate rate(s) and indicate subtotal. SIG Membership includes a Newsletter subscription.

Add Appropriate Overseas Air Option(s) to Voting/ Associate, Student or SIG Member Rate(s)

	Code #	ACM Members Voting/ Associate	ACM Members Student	SIG Mbrs Non-ACM Mbr. Rates	Partial	Full Air	Total
SIGACT (Automata and Computability Theory)	001	$15.00	$7.50	$40.00	+ $ 8.00	$29.00	= $
SIGADA (Ada)	037	20.00	10.00	42.00	+ 22.00	70.00	= $
SIGAPL (APL)	032	20.00	15.00	50.00	+ 9.00	31.00	= $
SIGAPP (Applied Computing)	042	15.00	7.50	39.00	+ 3.00	8.00	= $
SIGARCH (Computer Architecture)	002	28.00	14.00	54.00	+ 22.00	65.00	= $
SIGART (Artificial Intelligence)	003	15.00	8.00	41.00	+ 11.00	26.00	= $
SIGBIO (Biomedical Computing)	005	20.00	7.00	30.00	+ 6.00	16.00	= $
SIGBIT (Business Information Technology)	004	18.00	12.00	38.00	+ 6.00	11.00	= $
SIGCAPH (Computers and the Physically Handicapped-Print Edition)	006	15.00	6.00	42.00	+ 6.00	9.00	= $
SIGCAS (Computers and Society)	007	18.00	9.00	51.00	+ 7.00	10.00	= $
SIGCHI (Computer and Human Interaction)	026	30.00	10.00	52.00	+ 16.00	50.00	= $
SIGCOMM (Data Communication)	008	22.00	15.00	50.00	+ 10.00	40.00	= $
SIGCPR (Computer Personnel Research)	010	18.00	7.00	20.00	+ 7.00	20.00	= $
SIGCSE (Computer Science Education)	011	16.50	7.50	43.00	+ 12.00	44.00	= $
SIGCUE (Computer Uses in Education)	012	19.00	10.00	55.00	+ 10.00	20.00	= $
SIGDA (Design Automation)	013	15.00	15.00	27.00	+ 7.00	15.00	= $
SIGDOC (Documentation)	033	18.00	10.00	44.00	+ 7.00	11.00	= $
SIGFORTH (Forth)	039	20.00	11.00	46.00	+ 5.00	13.00	= $
SIGGRAPH (Computer Graphics)	015	26.00	16.00	59.00	+ 20.00	55.00	= $
SIGIR (Information Retrieval)	016	20.00	10.00	65.00	+ 6.00	37.00	= $
SIGLINK (Hypertext/Hypermedia)	043	22.00	12.00	56.00	+ 5.00	12.00	= $
SIGMETRICS (Measurement & Evaluation)	019	20.00	10.00	46.00	+ 9.00	33.00	= $
SIGMICRO (Microprogramming)	020	21.00	8.00	38.00	+ 5.00	20.00	= $
SIGMOD (Management of Data)	014	20.00	12.00	23.00	+ 16.00	55.00	= $
SIGNUM (Numerical Mathematics)	021	16.00	7.50	23.00	+ 8.00	16.00	= $
SIGOIS (Office Information Systems)	027	18.00	10.00	40.00	+ 10.00	35.00	= $
SIGOPS (Operating Systems)	022	15.00	8.00	41.00	+ 14.00	44.00	= $
SIGPLAN (Programming Languages)	023	25.00	10.00	52.00	+ 50.00	181.00	= $
FORTRAN FORUM (Fortran) Newsletter Only	038	10.00	6.00	20.00	+ 7.00	16.00	= $
LISP POINTERS (LISP) Newsletter Only	040	12.00	7.00	25.00	+ 6.00	16.00	= $
OOPS MESSENGER (Object-Oriented Programming Systems) Newsletter Only	041	10.00	6.00	20.00	+ 7.00	24.00	= $
SIGSAC (Security, Audit & Control)	036	18.00	9.00	44.00	+ 7.00	15.00	= $
SIGSAM (Symbolic & Algebraic Manipulation)	024	17.00	6.00	25.00	+ 8.00	12.00	= $
SIGSIM (Simulation)	025	15.00	5.00	41.00	+ 9.00	17.00	= $
SIGSMALL/PC (Small & Personal Computing Systems & Applications)	031	17.00	10.00	38.00	+ 8.00	17.00	= $
SIGSOFT (Software Engineering)	034	23.00	10.00	46.00	+ 17.00	60.00	= $
SIGUCCS (University & College Computing Services)	028	15.00	5.00	40.00	+ Free	Free	= $

SIG SUBTOTAL $ _____

Payment Information

Mail this application with your payment to:
ACM
P.O. Box 12114
Church Street Station
New York, NY 10257

ACM accepts payment by personal check, money order, or credit card. For international residents, payment must be in U.S. dollars drawn on a U.S. bank. Please make checks payable to ACM, Inc.

I wish to pay by: ☐ Check (make payable to ACM, Inc.) ☐ Credit Card

☐ American Express ☐ MasterCard ☐ VISA

Card #: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ Card Exp. Date: ____/____

Signature: _____

ACM Member Dues (from reverse side) $ _____

Publication Subtotal $ _____

SIG Subtotal $ _____

GRAND TOTAL $ _____

NEWSTRACK

FEDS BUST PLASTIC SCAM... The FBI broke a nationwide network of computer hackers reportedly making illegal credit-card purchases worth millions of dollars. Computer seizures were made in California, New York, Philadelphia, Seattle and Ohio. However, the number of participating hackers is feared to be over 1,000 strong, scores of arrests were still pending at press time. Hackers broke security codes, made charges on people's cards and created credit card accounts. The network also gained access to ATM and telephone account codes. The FBI's continuing investigation is focused in part on information the hackers reportedly obtained illegally from computers at Equifax Credit Information Services in Atlanta.

SUPER SUPPORTERS... Supercomputing ventures keep on clicking despite rumors of a shrinking market for multimillion dollar machines. Digital Equipment Corp. and Intel will collaborate on a range of software projects for massively parallel computers, the first of which is an architecture-independent programming language compiler that should be ready by the end of the year. Also, IBM has opened the Highly Parallel Supercomputing Systems Laboratory in Kingston, NY. The lab is designed to speed the process of bringing new parallel supercomputing products to market.

HELLO, GOODBYE... William Wulf, of the University of Virginia, steps into the role of chair of the Computer Science and Telecommunications Board this month. Wulf succeeds Joseph Traub, Columbia University, who launched the CSTB six years ago.

FRENCH STRATEGY... A group of research organizations in France are about to open an independent company to work with industry to translate discoveries into commercial products. *Nature* reports the new firm will offer government scientists full intellectual property rights and full royalties on licenses from patents based on their research in exchange for the laboratory paying all costs for the transfer of technology to the private sector. The French research establishment is expected to help this (still unnamed) company to gain credibility on the international scene.

VR GOES TV... Controversial filmmaker Oliver Stone (JFK) will coproduce the first TV show set in virtual reality. The six-part series, based on a popular cartoon strip called *Wild Palms*, is the story of a TV executive enmeshed in a "bizarre, abstract world where family relationships, corporate intrigue and murder all collide," according to an ABC spokesperson. The series may premiere this fall.

NEW HORIZONS... Two industry pioneers have set up shop to pursue a "vision of computing unlike anything that exists today." Microsoft's cofounder Paul Allen and former IBM vice president David Liddle are president and chief executive of Interval Research—a new R&D firm designed to focus on new ways of thinking about algorithms, communications and computing in the business world. One of Interval's first projects will be the creation of hand-held PCs that can tap into information databases worldwide.

CONTEMPORARY ART... An interactive museum exhibit created by a California artist asks visitors to take a walk in the shoes of a homeless person. Chris Hardman's "Etiquette of the Undercaste," which recently ended a run at the Smithsonian, is a computer-aided life cycle that begins in a baby's crib, and continues through reform school, street gangs and drug wars. By journey's end the participant is staring down the barrel of a gun, reflecting the death of the human spirit. The artist explains the exhibit as a way to help people understand how the homeless become homeless.

BANDING TOGETHER... An international consortium has been formed to promote and develop the ISODE package of open system interconnection applications used extensively in the research community. The primary goal of the consortium, which already boasts 11 founding organizations from the U.S. and Europe, is to spearhead key development of ISODE software at a faster rate than would be possible for any single firm. The U.S. office opened last month in Washington; a European office is expected to open by next spring.

RUSSIAN POLICY... A two-hour video conference between a delegation of Russian scientists and members of the U.S. House Committee on Science, Space and Technology emphasized that Russia's scientific community is looking for cooperation, not charity. "Although (U.S. assistance) could help alleviate our economic hardship, it could also provide knowledge that the U.S. now lacks," said Boris Saltykov, minister of science. The greatest obstacle, the delegation admits, is converting Russia's plethora of defense-minded scientists into commerce-conscious ones.

LEARN FROM EXPERIENCE... Chess-playing computers ignore their past successes and failures in favor of plowing through billions of possible moves. Robert Levinson, an assistant professor at the University of California, Santa Cruz, thinks forgetfulness is a great waste and is determined to teach a computer how to learn from experience. His goal is to build a computer system, named Morph, that mimics how a child would learn chess given only two elements: lists of legal moves and a playing partner who reveals only whether the child wins or loses. The National Science Foundation apparently agrees this concept has great potential for many areas in artificial intelligence technology; the agency has awarded Levinson a grant for more than $298,000 to build his Morph.

CHIP FIX... Scientists at the GE Research and Development Center have developed technology for designing "self-healing" chips. These next generation integrated circuits will police themselves for errors caused by malfunctioning circuit elements and produce signals that compensate for errors the faulted elements would otherwise cause. The technique lends itself to the design of both digital and analog fault-tolerant integrated circuits. GE researchers say the method is still two or three years away from its first real-life application.

BARRY'S BUG... Viruses, as we all know, can play strange and frightening games with computer-based data. Now, *Computing* magazine has reported a new strain that plays some strange, and yes, frightening music. It's called the Barry Manilow Virus—a phantom bug that's infiltrating a growing number of computer systems, scaring users with such tunes as "Mandy" and "Copacabana." The virus is a collection from the singer's *Greatest Hits* album. Once detonated, the virus spins out a continuous stream of Manilow's million sellers. Experts are working feverishly on an antidote for this plague. ■

PRESIDENT'S LETTER

June marks the last month of this administration and my last opportunity for a President's Letter. Needless to say, it seems like a good time to reflect on the last two years and access what we have accomplished.

First, however, I want to say it has been an honor and a pleasure to have the opportunity to serve as ACM president. My two-year term has provided challenges, rewards, frustrations, and enlightenment—sometimes simultaneously. I have certainly learned a lot, and I hope that in the process I have contributed something to the Association and to the profession and discipline of computing.

As I leave the presidency, I believe that, overall, ACM continues to be a healthy organization. Maintaining that health, however, has become an increasingly challenging task, as the effects of the ongoing recession impact the computing industry and, therefore, societies such as ACM. As measured by membership levels, subscription levels, technical activity, ACM is "holding its own" through this difficult economic period and, in that respect, is faring much better than many other organizations. I believe we have been able to maintain our membership and subscription levels because we have continued to aggressively promote the value of ACM membership and services. Moreover, through the efforts of Headquarters staff, we have increased the number of career-oriented services available to members. In reality, one should not be ashamed of flat growth during these economic times.

MOVING ON...

In addition to aggressively promoting ACM to maintain our membership and subscription levels, the Executive Committee has worked hard with Senior Headquarters staff and Senior ACM volunteers to reduce expenses when revenues have fallen short of plan. In the last fiscal year, over $1.3 million of planned spending against the General Fund (non-SIG) bottom line was eliminated. This year we are, once again, reducing expenses to ensure that the ACM fund balance remains at an appropriate level.

Despite having had to hold down spending as revenues slipped, we have still invested in new and important programs so that we continue to build the foundation for the programs, projects, and services that will ensure the future health and relevance of ACM.

One of our major investments during the past year was the move of ACM headquarters from 11 West 42d St to a new location at 1515 Broadway. In the process Headquarters left behind 23,000 sq. ft. of depressing space and moved into 33,000 sq. ft. of modern, quality work space. Additionally, all 95 members of the Headquarters staff have new workstations that are interconnected on local area networks to enhance cooperation and productivity. The new ACM Headquarters provides fine space and a quality working environment that the Executive Committee and Council believe will enable ACM staff to effectively support the activities of the Association well into the next century. Financially, it was an important step. We acquired 50% more space, which because of the soft real estate market

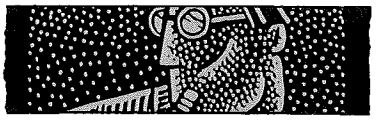
John R. White
ACM President

in New York, will cost substantially less than the old space. I encourage you to visit ACM Headquarters on your next trip to New York City.

Another major investment we are making is in local ACM activities. Bill Poucher, chair of the Local Activities Board, has assumed the challenging task of revitalizing local chapter, local SIG, and student chapter activities across the Association. Part of this revitalization effort has been the establishment of a new ACM Network Services program that will initially provide access to the Internet for volunteers who run local activities. The Executive Committee has found the money to launch this program and is presently considering additional funding, because many of us believe ACM Network Services has the potential to someday provide Internet access to *ALL* ACM members. The impact of this would be incredible. I would love to see the ACM renewal form with a box that says "check here to have an electronic mail box and internet address established for you at ACM.ORG."

We are also beginning to look seriously at electronic publishing. Proposals are being developed that define a staged, but major, program for electronic submission AND dissemination of information. The technology exists and the potential for enhanced electronic interconnection of ACM members through ACM Network Services really makes this an important decision for the Association. The main roadblock is money, and a special appeal is being made to you, the members, and other sources to build the ACM Development Fund to the point where this activity can be launched.

Additionally, we are continuing to invest in the important new directions for the Association that I outlined in the past. These include:
1. Increased public awareness of computing, as well as ACM;
2. Increasing engagement of issues important to the discipline, the field, and the public; and
3. Enhancing our role as an international organization.



Regarding the first of these, we completed, with your support, our commitment to the WGBH production of a five-part series on the history and importance of computing. Called *The Machine that Changed the World*, this series aired on PBS stations throughout the U.S. for five consecutive weeks in April.

Another activity aimed at increasing public awareness of computing was our continuing involvement with, and sponsorship of, the Computer Museum's Computer Bowl. The Bowl was held on May 1st in Boston and broadcast to many PBS stations as part of the *Computer Chronicles* series. As in years past, a number of questions used in the Computer Bowl were submitted by ACM members.

In addition to these public awareness activities, ACM is continuing to engage issues important to all of us. The ACM Committee on the Status of Women and Minorities in Computing was recently awarded a National Science Foundation grant to launch its mentoring program. The Education Board's task force on K-12 education is beginning to play an active role in understanding how to better use computing technology to improve precollege education. ACM Council itself has taken positions on issues regarding privacy and computer games and provided the National Research Council's Computing Science and Telecommunication Board position statements on difficult questions regarding the future of computing. Additionally, we are beginning to look at how to react more quickly as an association on important issues and how to responsibly take positions on US-specific issues without speaking for, or detracting from, our non-U.S. members.

Finally, we are continuing our efforts to be more effective in our role as an international organization. Although still in draft form, the current version of the FY 93 budget includes funds to open a small ACM office in Brussels.

None of this would have been possible without the dedication and hard work of ACM volunteers and staff. With that in mind, I would like to specifically and sincerely thank the members of my Executive Committee: Ron Oliver, Barbara Simons, Jack Esbin, and Bryan Kocher for their vision, dedication, and desire to "make a difference." It is also important to point out that our Executive Committee had a wonderful set of Board chairs with whom to work the past two years. Without their hard work and commitment, nothing would have actually been accomplished. Finally, within the volunteer community, I want to personally thank the ACM Council for their commitment to leadership. Council is becoming an important and distinguished body for the ACM and is increasingly willing and eager to engage and discuss issues, and to do so in a way that always attempts to forward action.

I also want to acknowledge and thank the entire staff at ACM Headquarters. ACM staff are real professionals and truly dedicated to seeing the Association achieve its potential. I especially want to acknowledge the support given to the Executive Committee by the executive director, Joe DeBlasi, and the deputy executive director, Pat Ryan.

Finally, I want to express my sincere appreciation to the Xerox Palo Alto Research Center for their support throughout my two-year term. Being ACM president can be time consuming—very time consuming. One cannot attempt to fulfill the responsibilities of this position without the full support of one's employer. In particular, I want to thank Mark Weiser, manager of the Computer Science Laboratory and my boss, and John Seely Brown, vice president and manager of PARC, for consistently conveying to me the value they attached to the investment I was making in ACM.

It's been great, and even fun —thanks!

ACM FORUM

THE CONTINUED DEBATE ON SOFTWARE PATENTS

In their hyperventilating "Viewpoint" (Jan. 1992, p. 17) Stallman and Garfinkle seem to have overlooked history and the literature in their haste to make an argument. In particular, in the section "Patenting What Is Too Obvious To Publish," they mention the windowing techniques of the MIT Lisp machine, comparing it favorably with mechanisms developed by Rob Pike. They impute to the Lisp machine developers a belief that their work was "too obvious to publish."

Perhaps it was; in contrast, Pike's work was published, and was very well received. His presentation in SIGGRAPH 1983 was one of four papers selected for publication in *TOG* [7]. Its review in *Computing Reviews* [5] concludes, "This is a well-written, comprehensive paper on a very popular subject, and will doubtless become required reading for designers of window management systems."

Subsequently, Foley, vanDam, Feiner, and Hughes [1] described it as follows: "In a seminal paper, Pike... described [window management] on a bitmapped screen... . The ideas of this system are now seen in the X Windows System, the Lisa operating system, the Macintosh operating system, and many other windowing systems."

Pike's published work was evidently important and influential, while the Lisp Machine developers' work was either unpublished and thereby withheld from the scholarly community, or as obvious and unimportant as Stallman and Garfinkle claim. Neither possibility supports the thesis of their "Viewpoint" article.

Dennis M. Ritchie
AT&T
Murray Hill, NJ

The League for Programming Freedom brings up the issue of software patents in the U.S. This issue is discussed from an American point of view, and I think it should be extended to an international approach. I agree with the League about the fact that abusive software patents can be a danger for the software industry as a whole, but the impact on the exchange of software between the U.S. and the rest of the world may also be greatly affected.

The League mentions the high price of patent searches. This price is even greater for foreign companies, who cannot readily get this information. Costly patent searches could very easily become prohibitive for a company which makes only a relatively small part of its revenue in the U.S. If patent suits become commonplace, as we can fear, these companies might simply have to withdraw from the U.S. market. In some fields, like constraint logic programming, European companies represent a large share of the market and research. The withdrawal of these companies from the U.S. market would not only be a great loss for the individual companies, but also for the entire U.S. constraint logic programming community, which would then have trouble staying informed of the recent developments in the field.

On another level, numerous suits against foreign companies trying to sell software in the U.S. could easily be taken as a protectionist measure from the U.S. by the governments, which would then be tempted to impose restrictions on U.S. software import. Even though this scenario is somewhat far-fetched, its eventual impact is something we have to consider today.

Software patents can not only "snuff out the sparks of creativity of

individualism that have driven the computer revolution" in the U.S., but also block the exchange of software and ideas between the U.S. and other countries, which is another important input to this computer revolution around the world.

Eric Vètillard
47 Boulevard Aguillon
Marseille, France

I am in agreement as to the source of the problem. 35 U.S.C. 103 rejects patentability if it "would have been obvious... to a person having ordinary skill in the art to which said subject matter pertains." Prior to 1991, the PTO refused to hire CS graduates, nor did it allow CS graduates to practice before it. The result was the blind leading the blind. Neither practitioners nor examiners had any idea what was "obvious" since they did not have an "ordinary skill in the art."

Additionally, the PTO has used its lack of ability to find relevant art as an excuse against patentability of software. Examiners and practitioners have not belonged to the ACM, nor have they shared access to such informal sources of information as the USENET. Also, it is partially a problem of not having the shared lore acquired through a CS degree or long practice in the field.

My first disagreement with the League is in its implication that if a pair of programmers could write 50,000 lines of code in a year, much of that code could possibly be patentable. Little, if any of it would be. First, the only way for two persons to design, code, and debug that much code in one year is if most of it is "obvious," like what COBOL programmers do.

Secondly, most software patents being issued, and almost all such patents upheld, concern operating system or hardware-oriented software. This type of software can be measured in the hundreds or low thousands of lines of code per programmer year. First, this type of code has enough novelty for patentability. Secondly, patents cost money; software patents cost a lot more money than most patents. Finally, the Supreme Court in a line of cases has severely restricted what software is patentable.

Justice Douglas, born in the last century, wrote the *Gottschalk vs. Benson* decision in 1972. It stands for the proposition that computer programs are algorithms, and that mathematical algorithms are not "statutory" under § 101 because they are rules of nature. This would be like patenting gravity. This is the law today, only slightly weakened by *Parker vs. Flook* and *Diamond vs. Diehr*. As all concerned agree, it is also ridiculous. In its recent decisions, the CAFC has recognized the reality of software patents, requiring for patentability some hardware component, or that the software be part of some machine (other than as a program in a general purpose computer).

The authors seem to think that the PTO has the last word in patentability. Patents are invalidated every day. Yes, patents are "presumed" valid. However, this presumption applies only to questions of fact. It does not apply to questions of law, such as whether a patent concerns statutory subject matter.

Even if a patent is valid, it is often not enforceable. Patent infringement suits are horrendously expensive, especially for the plaintiff. One of the litigators in our office uses $2,000,000 in damages as a rough yardstick for taking a patent infringement case on a contingency basis. The authors use a 5% royalty figure. We have seen what happened to IBM's MCA royalty offer. A reasonable royalty of 1% is probably even generous. Note the .8% royalty rate for use of the FAX compression algorithm. A 1% rate results in the requirement that any infringing product have sales in excess of $200,000,000 before it is economic to sue for patent infringement. Companies with that sales volume are exactly those that can cross-license.

Clients in any patent law office receive constant cease and desist threats alleging patent infringement. We usually advise our clients to ignore the threats, often writing an opinion letter as to the invalidity or unenforceability of the patent claimed. Because of the cost, we know that patent holders will almost never actually sue for infringement. One of our attorneys, after practicing for 25 years, and after seeing hundreds of demand letters, has seen two clients sued for patent infringement. Both suits settled.

One additional problem is that of proof. The more obscure the patent, the harder it is to show the good faith belief in infringement required for suit without access to source code. Have you ever tried to deduce an algorithm from disassembled object code? Source code is usually not available until *after* a suit has been filed. Without a good faith belief in infringement at the time a suit is filed (supported by facts available at that time), any party suing for infringement is open to Rule 11 sanctions. Every day Federal courts become more willing to assess attorneys' fees (in the case of patent infringement, costing upwards of $200,000) against plaintiffs (and their attorneys) in frivolous cases.

My final criticism of the League's article is the authors stated preference for copyright law. Traditionally, patents have protected ideas, and copyrights have protected the expression of ideas. Copyright law has expanded to fill the vacuum in the software field created by the traditional unpatentability of software. Note the *Paperback Book* decision, as well as the ongoing Lotus and Apple "look and feel" copyright suits.

Copyright law appears to me to

be a much greater threat to software freedom that patent law: it is free; it is automatic; it lasts almost forever (life of author plus 50 years); infringement is easier to prove (validity is trivial), and remedies and damages are invariably more generous (for example, it is much easier to recover attorneys' fees).

Bruce E. Hayden
Fields, Lewis, Pittenger, and Rost
Patents, Copyrights and Trademarks
Denver, CO

Thank you for the public service you have done by publishing the League for Programming Freedom's "Viewpoint."

The League did a good job of explaining the problem and proposing a common-sense solution. If I may, I'd like to emphasize the following three points:

1. To define software for purposes of the proposed legislation against software patents, the League emphasizes the mathematical aspects of software, and the ease with which it can be copied. I think it is also important to distinguish software inventions clearly and unambiguously from inventions that might properly be patented, as follows:

A software invention is an expression of instructions for arranging and operating a tangible device. In that respect, it is like a wiring diagram, instruction manual, blueprint, or specification, except that devices can interpret software directly. Like diagrams, etc., software is not part of the device for which it provides information. No software can ever provide, store, use, or transfer energy in any form, so it can never be part of a device. A software invention is nothing more nor less than the expression of an idea.

2. Many software firms have been founded on the sole promise of reinvented, incrementally improved, or new software inventions. Some of them are now industry leaders, like WordPerfect Corporation. If software patents are permitted to continue, there will be no more success stories in our country like that of WordPerfect or Lotus Development.

3. Software patents harm computer science education, since teachers and students distribute thousands of programming algorithms, which are supposed to be in the public domain. Fear of patent suits will force colleges and universities to stop the free exchange of computer science ideas expressed in computer programs now transmitted via vast international computer networks. Software patents will succeed in doing what computer viruses could not. Surely the Founding Fathers did not imagine that the Patent Office might be so misused.

Sidney L. Sanders
Sanders-Intek, Inc.
Geneseo, NY

Response

To Dennis Ritchie: The idea of backing store is simply that the hidden parts of a window can be held, and updated, in off-screen memory.

If Rob Pike had had nothing to say beyond this, his paper would hardly be noteworthy. The real meat of the paper was the memory management scheme which minimized the amount of off-screen memory needed by subdividing it into many rectangles of different sizes. Pike designed this scheme to support a terminal with limited memory space.

The Lisp Machine developers did nothing so impressive. With virtual memory and 80 megabytes of swapping space, they did not need such complexity. Their main concern was to minimize changes to the screen-drawing microcode.

Pike's memory management scheme was a clever piece of design, but it is not very important in window system design today. This is because the decreasing price of memory has removed the need to try so hard to minimize memory use. X Windows, in particular, uses a much simpler scheme.

If the AT&T patent were limited to Pike's memory scheme, we would still impose obstacles in improving window systems, but it would not fall in the category of "too obvious to publish." It would also be no threat to the users of X Windows. However, the patent covers any implementation of backing store that handles multiple client processes, whether complex or simple. Pike's scheme was the occasion for the patent, but it has little to do with the effect of the patent; that derives from the patent's broadest claims. These are what we are talking about when we say that the matter is obvious.

In response to Eric Vetillard: We do not know of a particular trend to sue or threaten foreign companies now. However, there is some sign that officials in Washington regard extended intellectual property as a means to improve U.S. balance of payments, completely regardless of details or of the other effects of the policy.

Following this view, the U.S. has repeatedly pressured other countries to adopt software patents. For example, the U.S. has put provisions in the draft GATT treaty that would require all countries to have software patents. The often-suggested idea of making software patents last for a time commensurate with the rate of progress in software is also prohibited. So is any sort of mandatory licensing scheme that might lessen the practical burden of the patent system. Thus, Washington's response to the effect of patents on competitiveness is to spread similar difficulties elsewhere.

However, it should be noted that the usual assumption that Americans would own the bulk of any

newly imposed software monopolies is less and less true. Over one-third of the U.S. software patents being issued today seem to be going to Japanese companies.

In response to Bruce Hayden: Hayden has a good point that patent lawsuits on a contingency basis would have to be made against large companies. However, this does not mean that only large companies need fear.

American Multi-Systems is a small family business that recently developed a game system for bingo halls. It is in danger from Fortunet, which has a broad and simple patent on using a network to implement casino games, and has already sued another developer named CPU Technologies. Fortunet offers no licenses because it aims to have a monopoly on such systems. Though American Multi-Systems has not yet been sued, it finds that many customers avoid it because they fear that they too might be sued.

A small expenditure is enough to crush someone who cannot afford to resist. Phil Zimmermann put RSA encryption together with various other algorithms to produce a free program called Pretty Good Privacy. This program is now widely used outside the U.S., but it is unavailable here; Zimmermann agreed to stop distribution when threatened by RSA, Inc., which claims its patents cover all public key encryption.

Hayden underestimates the abilities of capable system developers when he says that a 50,000-line program written by two programmers in a year must be trivial. But he is probably right that the program would contain at most a few new ideas that could be patented.

However, this does not protect the authors from the danger that many of the techniques they use—perhaps well known, or "obvious" in the opinion of software designers—may have been previously patented by others.

If it were true in practice that patents required some special hardware component, and could not apply to pure software, then this might solve the problem of software patents (depending on details, of course). In practice, as Pamela Samuelson explained in *Communications* (Aug. 1990, pp. 23–27), this requirement has been interpreted by the courts to mean that a patent applicant need only mention a ROM to satisfy it.

Thus, for example, we have patent 5,031,134 which covers an algorithm for integrating a function from a table of values stored in a ROM.

If this and similar patents applied only to machines using ROMs in this way, they would be meaningless. There is little call for a device that integrates the same function each time it is used. But, through the doctrine of equivalents, the patents may also apply to programs running on general-purpose computers.

As long as the court of appeals continues with this interpretation of the law, there is little scope for a defendant to challenge a patent's validity except on questions of fact.

If it becomes harder to subpoena copies of source code, this will protect software developers from certain patents, provided they do not release source code. It is ironic that the patent system should work in this fashion to encourage trade secrecy—the opposite of its original intent.

Note, though, that this factor would not apply to the patents listed in our article. Some apply to the user-level features of a program. Even when a patent covers techniques or algorithms, it is often possible to tell which ones are used by looking at what the program does.

The League for Programming Freedom
1 Kendall Sq. #143
P.O. Box 9171
Cambridge, MA 02139

The Caller ID Debate

The "From Washington" column, "Privacy in the Telecommunications Age" (Feb. 1992, p. 23), is embarrassingly facile, almost insultingly so.

The authors of this column are Carol Wolinsky and James Sylvester, lawyers in the employ of Bell Atlantic. Bell Atlantic is a principal antagonist in the current debate over personal privacy and new telephone technology. Nowhere in the column is Bell Atlantic's aggressive advocacy of privacy-invading technologies even acknowledged, let alone explained. Instead there are coy little commentaries strewn about, like "the advent of extensive databases coupled with easy-to-use, menu-driven search and retrieval tools will continue to drive our nation's economic development." One pleads for some empirical support for this baldly self-serving assertion, whose other outcome may be a substantial diminution of our personal liberties.

The authors ingenuously claim that the man objection to Caller ID, which they champion the loudest, is "that the service violates the privacy rights of the calling party by disclosing his or her telephone number." This is only half of the problem. More to the point of the ACM's concerns, civil libertarians worry that Caller ID will lead—and is intended to lead—to the computer-abetted collection, manipulation, matching, and resale of personal information, with the telephone number providing a convenient header to our personal dossiers. That is why the telephone companies, including Bell Atlantic, so vociferously argue that Caller ID must be universal, with no easy blocking options. Otherwise, it will have only limited commercial appeal. After years of fighting imposition of government-supplied national ID numbers, we turn around to find the telephone companies providing them instead. Many of us are alarmed.

Communications should either

find genuinely neutral parties to argue important social issues, or balance presentations such as this flyweight brief with opposing points of view (of, one hopes, higher standards).

Robert Jacobson
Former Staff Director, Assembly Utilities and Commerce Committee
California Legislature

The column by Wolinsky and Sylvester of Bell Atlantic contained several inaccuracies pertaining to the so-called Caller ID telephone service.

Wolinsky and Sylvester state, "This service provides the called party with the directory number of the calling party and enhances the privacy of call recipients because it allows them to identify the caller in advance." All clauses of this sentence are false. The service does *not* provide the directory number of the calling party, it provides the number of the telephone that the calling party happens to be using. People (friends, relatives, coworkers, and harassers) call from a variety of numbers (home, office, other people's phones or offices, phone booths) so the service does not identify callers reliably. Its value to call recipients in enhancing their privacy is therefore marginal. Are users supposed to memorize the number of all the places where all of their friends might call from? How are users to know whether an unrecognized number is a telemarketer or a spouse calling from a phone booth with a broken-down car?

The name for the service used by most telephone companies, Caller ID, is misleading. A few telephone companies use a more accurate and honest name, "Calling Number ID" (CNID).

Though CNID is poorly designed for use by residential phone users in screening calls, it is quite useful for businesses in collecting marketing data. Businesses fully intend to use CNID to collect the numbers of households that call them and, with the help of reverse directories and database enhancement services such as those provided by Telematch and Donnelly Marketing, compile complete marketing databases on American households.

Phone companies could, if they wanted to, design services and equipment that would be more useful to residential phone users than CNID is for enhancing the privacy of call recipients, and less helpful to businesses for collecting marketing data. For example, phone companies could provide a call-screening service: differential call-handling for lists of numbers you specify, e.g., family, friends, foes, other. Types of call-handling could include taking a message, ignoring the call, ringing normally, ringing distinctively. Such a service would not disclose anyone's number without their consent.

Another CNID-alternative is to identify callers by billing name or by an optional text string provided by the caller at the time of the call, e.g., "Grandma". Both approaches have the important advantage of not using identifying information that is conducive to compiling and searching databases. The latter approach has the additional advantage of truly identifying callers, not just calling telephones, making it much more reliable for call-screening than CNID is.

In proposing CNID, phone companies are clearly pandering to the needs of their business customers and neglecting those of their residential customers. In California, citing testimony from Computer Professionals for Social Responsibility and others, a judge for the Public Utilities Commission recently suggested that CNID be disallowed in that state on the grounds that it violates people's right to control the disclosure of personal information, and called upon the telephone companies to develop services that are more useful to residential phone users.

Yes, we want better call-screening capability. No, we do not want Calling Number ID.

Jeff Johnson
Computer Professionals for Social Responsibility

Response

Johnson's and Jacobson's letters in response to our column are indicative of the fervor and debate surrounding privacy issues. While Johnson chooses to focus on Caller ID issues, Jacobson's letter discusses a broader range of issues—but misses the point of our column. We clearly stated our purpose at the outset—to discuss some of the current issues where technological progress is raising concerns about personal privacy. Many people remain unaware of even this limited range of issues being considered by the U.S. Congress, the FCC, and the states as they formulate telecommunications and privacy policy.

It is true that Bell Atlantic has been an advocate for Caller ID service. But, our discussion pointed out the diversity of views and the varying treatment of this service by 18 states, Congress, and the FCC without promoting Bell Atlantic's views in particular. Bell Atlantic continues to believe that unrestricted Caller ID, meaning, offered without the ability to block the display of the calling number, is the best deterrent to annoying and harassing calls. In our column, we choose not to extol the virtues of Caller ID, but the service is very well received by our residential customers; they represent over 93% of our subscribers. Small business, schools and public safety personnel also gain significant benefits as false orders and bomb threats have been significantly reduced. Anyone, contrary to Jacobson's assertion, can block display of their telephone

number by placing the call through an operator or by using a payphone or a credit card.

Jacobson's and Johnson's assertions that telephone numbers will become national ID numbers and provide the linchpin for building personal dossiers ignore the facts. Personal information is already widely available and subject to far fewer safeguards than the telephone companies impose on release of their data. There is nothing magic about telephone numbers in comparison to the role played by an address and zip code or a driver's license number, social security number, or credit card number.

However, telephone numbers can meet customers' expectations for new, privacy-enhancing services. The emerging Personal Communications Services are a case in point. Customers want to initiate and receive calls from any place, at any time, but not necessarily from anybody. The advanced technology that supports Caller ID makes these services possible and already provides many of the services proposed by Johnson.

Jacobson pleads for "some empirical support" for our statement that extensive databases and menu-driven search and retrieval tools will have a major role in U.S. economic development. We made this statement in our discussion of the debate surrounding telemarketing practices; support for our summary statement is readily evident in business news reports.

Finally, our employment by Bell Atlantic was clearly acknowledged. We are not lawyers, as Jacobson believes, but we find, sometimes regrettably, that you cannot deal with these issues in Washington without unintentionally sounding like one.

Carol Wolinsky
Manager, Privacy Issues

James Sylvester
Director, Infrastructure Issues
Bell Atlantic

Optimization-based vs. Rule-based

We read with interest "A Rule-Based Solution for Dense-Map Name Placement" (Jan. 1992, p. 68) by Doerschler and Freeman. Reading that article led us to reconsider the article, "Integer Programming vs. Expert Systems: An Experimental Comparison" (Mar. 1990, p. 323) by Dhar and Ranganathan. Both articles present rule-based solutions to combinatorial optimization problems. Dhar and Ranganathan compare optimization-based and rule-based programs for the generation of college course schedules. Doerschler and Freeman address map name placement. Both articles ignore previously published solutions to similar problems using mathematical optimization techniques.

During a half-day search through journals devoted to operations research we were able to find one review article [10] and five application-oriented articles on course scheduling, classroom assignment, and the assignment of teachers to classes [2, 3, 4, 6, 8]. Each application article formulates its particular version of the problem as a combinatorial optimization problem and describes custom coded optimization programs developed to solve them. All five programs use some combination of problem partitioning, constraint relaxation, problem specific heuristics, and user intervention to help find feasible solutions. Two of the programs have been used by colleges to solve real scheduling problems.

Doerschler and Freeman allocate exactly two sentences in their article to the possible use of mathematical optimization techniques to solve map name placement problems. In the second sentence they dismiss the possibility for dense-maps because of the size of the optimization problems involved.

In fact, the two map name placement programs described in the cartographic literature *which are used today in map production* rely on optimization techniques to produce high-quality output. These two name placement programs are designed to assist in the compilation of special purpose maps. One program places area feature names inside map feature polygons [9]. The other places point feature names on petroleum industry basemaps [11, 12, 13]. The fact that these two programs are used for special purpose mapping is less important than the fact that they are used in map production, something that cannot be said for any rule-based map name placement program. The petroleum industry-oriented program was implemented by one of the authors of this letter, and he can assure readers that petroleum industry basemaps often fit into the category of maps with "dense name placement."

We do not assert that optimization-based programs are intrinsically better than rule-based systems for resource allocation in the university environment or for map name placement. Dhar and Ranganathan certainly point out valid limitations in the use of optimization algorithms for course scheduling. But to be fair to readers, authors should cite articles which might provide alternative solutions to those presented in their own articles. In the case of these two publications, the authors should have provided references to articles which solve similar problems using optimization techniques.

Steven Zoraster
Landmark/Zycor Inc.
Austin, TX
Ronald Sawey
South West Texas State University
San Marcos, TX

References

1. Foley, vanDam, Feiner, and Hughes. *Computer Graphics: Principles and Practice.* Second Ed., Addison Wesley, Reading, Mass. 1990.

2. Glassey, C.R. and Mizrach, M. A decision support system for assigning classes to rooms. *Interf. 16,* 5 (Sept.–Oct. 1986), 92.

3. Laporte, G. and Desroches, S. The problem of assigning students to course sections in a large engineering school. *Comput. Oper. Res. 13,* 3 (1986), 387.

4. McClure, R.H. and Wells, C.E. A mathematical programming model for faculty course assignments. *Dec. Sci. 15,* 3 (1984), 409.

5. McPherson, A.L. Review 8408-0671. *Comput. Rev. 25,* 18 (Aug. 1984), 390.

6. Mulvey, J.M. A classroom/time assignment model. *The Europ. J. Oper. Res. 9,* 1 (Jan. 1982), 64.

7. Pike, R. Graphics in overlapping Bitmap Layers. *ACM Trans. Graph. 17,* 3 (Jul. 1983), 331.

8. Tripathy, A. School timetabling—A case in large binary integer programming. *Manag. Sci. 30,* 12 (Dec. 1984), 1473.

9. van Roessel, J.W. An algorithm for locating candidate labeling boxes within a polygon. *The Amer. Cart. 16,* 3 (Jul. 1989), 201.

10. White, G.P. A survey of recent management science applications in higher education administration. *Interf. 17,* 2 (Mar.–Apr. 1987), 97.

11. Zoraster, S. Integer programming applied to the map label placement problem. *Cart. 22,* 3 (1986), 16.

12. Zoraster, S. The solution of large 0–1 integer programming problems encountered in automated cartography. *Oper. Res. 38,* 5 (Sept.–Oct. 1990), 752.

13. Zoraster, S. Expert Systems and the map label placement problem. *Cart. 28,* 1 (1991), 1.

High marks for ACM's COMPUTING ARCHIVE from the people who know CD-ROMs *best*–

From *CD-ROM Professional:*

"There is no other product on the market that specifically targets the research computer scientist or engineer. ***Computing Archive*** *provides access not only to the publications of ACM, which is the major publisher of scholarly computer literature, but to literature from all the technical publishers. Academic libraries with large computer science departments should consider purchasing this database. Users will be very enthusiastic about the full-text Computing Reviews and the page images."* (January 1992)

Computing Archive, a CD-ROM database, contains a decade's worth of bibliographic citations of the most significant theoretical and scholarly work in computer science. As well as easily-searched bibliographic citations of the most important professional computer literature published, users get keyword descriptors for each entry and full-text reviews of articles and books covered by *Computing Reviews*, ACM's pre-eminent review journal.

Update

The updated ***Computing Archive*** is available as of July 1992. Computing literature citations from over 450 journals, hundreds of conference proceedings and all the major technical publishers' books beginning in the early 1980's is now updated through June 1992.

Enhancements

Improved search and retrieval software for the update brings users the following new benefits:

1. Cut Text, a feature which allows you to print out or export, in any order, bibliographic items or reviews that you want to retain in hard copy or your personalized reference file.

2. Publisher AddressBook, an invaluable compilation of publishers of journals and books, with their addresses for easy full-text ordering.

3. Windowing options for viewing found records next to full text or next to page images.

4. *Computing Archive's* User Guide and on-screen help is fully revised, to make it even easier for new or infrequent searchers to use this highly intuitive interface. A separate section on advanced search techniques helps experienced users and library professionals mine this database to the fullest.

Ordering Information

New Subscribers:

Nonmembers	$1,299.00
ACM Members	$999.00

Renewals:

Nonmembers	$959.00
ACM Members	$659.00

Charter subscribers are entitled to 5% off list price on the renewal; contact ACM Manager of Technical Publications for more information, and to discuss network pricing.

Orders or further information from:

ACM CD-ROM Publications
1515 Broadway, 17th Floor
New York, NY 10036
(212) 626-0667 or -0668

acm PRESS

Minimum configuration: IBM XT/AT or 100% compatible with 512K memory, 1 floppy drive, 1 CD-ROM drive, VGA monitor highly recommended, PC DOS or MS DOS 3.0 or higher, Microsoft Extensions. Printer for bit-mapped page images suggested. CA.CACM 6/92

TECHNICAL CORRESPONDENCE

A CRITIQUE OF THE PROGRAMMING LANGUAGE C*

The data parallel programming language C*® was originally developed for the Connection Machine. Efforts are now underway to standardize a revised version of C* [7]. We think that standardization of C* is premature at this time, since the language contains a number of unproven constructs and obvious flaws. We are concerned that standardization of a parallel language now might force its programming model upon future generations of programmers, *even though we already know it is deficient.* The purpose of this note is to make the relevant issues accessible to a wider audience and to make specific recommendations for improving C*.

C* is an extension of ANSI standard C and intended as an "efficient, fairly low-level systems programming language" [7] for parallel computers with distributed memory. Parallelism is expressed directly in the data parallel paradigm. In this paradigm, "parallelism comes from simultaneous operations across large sets of data, rather than from multiple threads of control" [2]. Data parallelism is a synchronous paradigm and therefore well suited to SIMD machines. It has also been implemented successfully on a MIMD machine [6].

As an extension of C, C* inherits most of the drawbacks of its ancestor, but we are not concerned about those here. Neither are we concerned with limiting C* to a synchronous paradigm, even though an asynchronous one would be more general. We are concerned, however, with the principles of programming language design, the programming model underlying C*, and the efficient implementability of C* on both SIMD and MIMD machines. The problems we identified in C* in these areas are discussed in this article.

Parallel Data Types

C* introduces the parallel variable as a new data type. The parallel variable is an array of one or more "parallel" dimensions. All elements of a parallel dimension may be processed simultaneously, while the traditional, "serial" dimensions can only be processed serially. For example, if one wishes to define a two-dimensional array *A* whose rows can be processed in parallel, but whose columns are processed serially, then one declares the following:

```
shape [N]rowdim;
float:rowdim A[N];
```

The *shape* declaration given here introduces *rowdim* as the name for a storage structure that can hold variables with a parallel dimension of *N* elements; note the index range declared on the left of the identifier. The second declaration then allocates a variable *A* with the shape given by *rowdim*. The elements of this variable are again vectors, but with a serial dimension. This time, the index range is on the right. The left and right indexing carries over into accessing arrays: The expression [*i*]*A*[*j*] would select the element in the *i*-th row and *j*-th column of *A*. This notation is unusual, but is intended to provide syntactic clues to the programmer about which dimensions can be processed serially

and which in parallel. A minor annoyance is that if the programmer should decide to change a dimension from serial to parallel or vice versa, all index expressions in the program involving the changed type must be switched around accordingly.

While syntax is a matter of taste, nonorthogonality of the new type constructor for parallel variables is a more serious problem. Parallel variables cannot be combined freely with other types. For instance, it is not possible to create parallel variables with records as elements that in turn have parallel variables as components. Perhaps one could argue that this particular restriction is minor for the intended application area of C*. A more serious nonorthogonality concerns pointers. Pointers may not be stored in parallel variables (or in records or other data structures stored in parallel variables). This restriction is unfortunate, since there are many non-numeric applications that could use parallel pointers [1]. Also, omitting them is inconsistent with the spirit of C, where pointers are used frequently. The prohibition against parallel arrays of pointers seems to be motivated partly by the addressing properties of the Connection Machine, and partly by the type structure of C* itself. The earlier version of C* has about ten different variants for each pointer type. The variants reflect whether the pointer itself is stored in a singular or a parallel variable and whether it actually points to a singular or parallel variable, plus some additional variants. The result is that parallel pointers in old C* are exceedingly complicated to program. It appears that the same complexities would arise in new C*, and were omitted for this reason.

The source of these difficulties is quite simple: The orthogonal notions of data type and data layout have been intermixed in C*. A data type determines which operations can be applied to a datum; the layout determines whether the operations can be applied in parallel or serially. These are two separate, independent notions. The set of *what* operations can be applied to a datum should be independent of *how* (in parallel or serially) they can be applied. A better approach appears to be to rearrange a variable of a given type implicitly and automatically to fit a required layout (perhaps with a performance warning from the compiler).

No Nested Parallelism

Nested parallelism occurs when a parallel program calls a procedure or another statement which spawns additional parallelism. This feature has been found to be necessary for writing high-level parallel programs [1]. Nevertheless, nested parallelism is not possible in C*. Instead, the nested parallelism must be pushed up to the top level of the program. This property forces programmers to distort otherwise clear programs, prevents the topdown structuring of parallel programs with subprograms, and hinders reusability. We will illustrate these problems with a somewhat longer example.

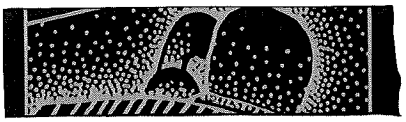
Suppose we wish to write a program for searching large game trees, such as in chess or checkers. The nodes in the game tree are board positions, where an edge connects two nodes if a single, legal move leads from one node's position to the other's. The task is to write a program for expanding the game tree from a given position down to a certain level. This situation is typical for many search algorithms where the search space is irregular and cannot be given a priori. A clear, recursive outline for building and searching the tree is as follows:

```
void SearchTree(position p) {
 if (p->depth < maxdepth) {
  successors=GenerateMoves(p)
  forall i in len(successors) do
   SearchTree(successors[i])
}}
```

The function *SearchTree* obtains a single board position as parameter and tests whether the maximum search depth has been reached. If not, it calls function *GenerateMoves,* which generates the list of legal successor positions and returns it. *SearchTree* then spawns as many additional invocations of itself as there are successor positions. Note that there may be many simultaneous invocations of *SearchTree* operating simultaneously, but under a synchronous paradigm, they will all perform their various actions, such as calling subprograms, in perfect synchrony. *SearchTree* would also run unchanged under an asynchronous paradigm.

While this program can be transformed to fit C*, the result is not nearly as clear and concise. First, *SearchTree* and *GenerateMoves* must be changed to accept a vector of positions as parameters. Second, *GenerateMoves* must be split into two parts. The first part estimates the number of successors for each position in the vector. These numbers are added and the result is used to allocate a new position vector long enough for storing all successors of the input vector at once. The second part of *GenerateMoves* then fills in the successor positions. Finally, the filled vector is passed to another invocation of *SearchTree.* This transformation illustrates how the parallelism inside *GenerateMoves* and *SearchTree* has to be marshaled and pushed up to the calling procedure. Although this transformation is not particularly hard to program and becomes easier with practice, we believe it would be better performed by the compiler and runtime system. Note also that after the transformation, the idea of multiple threads operating simultaneously has been lost, and a compilation of the transformed program for an MIMD machine might be inefficient due to the forced synchrony.

The non-nested parallelism of C* is inappropriate for writing clear, maintainable, and portable



parallel programs. One might argue that it is still unclear which forms of nested parallelism are appropriate and how to implement them, but that is precisely our point: It is too early to standardize nested parallelism out of existence with C*.

Multiple Copies of Each Function

For scalar functions, the programmer must write two versions: one for parallel, the other for sequential contexts. For instance, suppose that we have written a scalar function *abs(x)* that returns the absolute value of an integer. It would be natural to apply *abs* to parallel variables *v*1 and *v*2 of shape *rowdim* thus:

```
with(rowdim) v2 = abs(v1);
```

The **with**-statement in C* activates as many (virtual) processors as there are elements in the given shape. These processors operate on the given parallel variables elementwise. This notation is normally used for all scalar operators and assignment. However, the presence of the call to the scalar function *abs* makes it illegal. To make it legal, the programmer must write a second function *abs* that takes a parallel variable of shape *rowdim* as parameter. Thus, the programmer must write at least two versions of each function. Additional versions are needed for additional shapes, or shapes must be passed as parameters and a case analysis performed inside the function. The difficulties of keeping multiple versions of the same function consistent are well known to practicing software engineers.

This discussion points to a problem with the semantics of the **with**-statement. **With** creates multiple processors that operate in parallel, but when they reach a function call, only a single call is actually performed. Inside the function, however, the original processors come back to life. Thus, the parallel context seems to be conceptually suspended for the moment of the call, then resumed inside the procedure. Apparently, the specifics of the procedure call on a SIMD machine are reflected in the language definition. A better, fully consistent view would be to let the **with**-statement create as many processors as before, but let all of them execute the call of the (scalar) function. Since the processors operate synchronously, there is an efficient implementation even on a SIMD machine. A separate, parallel version of each function need not be written. These simplified semantics also accommodate nested parallelism.

Control Structures

The control constructs for loops and conditional statements are defined in an awkward way and fully synchronous execution may be too restrictive for efficiency. As an example, consider the following parallel loop in C*.

```
while (|=(⟨parallel-condition⟩)){
  where (⟨parallel-condition⟩){
    ⟨statements⟩
  }
}
```

The intent is that multiple processors execute the above loop simultaneously. The whole statement terminates as soon as ⟨parallel-condition⟩ evaluates to false in all processors. The operator |= in the first line, an OR-reduction, expresses this termination. It is awkward to be forced to repeat this condition in the second line. The careful programmer would evaluate the condition only once and then store it into a temporary, in order to prevent unwanted side effects and inefficiency. The repetition could easily be avoided and the compiler could provide the additional code. At least this is how it was in the original C*.

The language definition as it stands also hurts performance on MIMD machines. The problem is the overly synchronous behavior required for loops: all iterations execute in complete lockstep, even if each loop operates on purely private data. This behavior is acceptable on a SIMD machine since the hardware forces that behavior anyway. But on MIMD machines this could hurt performance. Suppose the loops were allowed to run asynchronously, then some "natural" load balancing might occur. That is, suppose one processor executes the first iteration quickly and the next more slowly, while another processor exhibits the opposite behavior and thus the two processors finish in about the same time. With the present language definition, the processors are forced to run fully synchronously, and hence are slowed to the speed of the slowest one.

Conclusion

There are many other, small problems in C*, which we will not discuss further. (Among those are (1) that value parameters of only a single shape can be passed to functions, (2) that `a += b` has not necessarily the same effect as `a = a + b`, (3) that vector operations are defined for parallel dimensions, but not serial ones, and (4) that the language is defined mostly by example and not by a precise statement of the semantics.) While we consider the old C* a first and significant step in the right direction, it is dismaying to see so many of the old problems being carried over into the successor. It seems that elementary principles of language design such as machine-independence, orthogonality of constructs, consistency, and simplicity have not been taken into account sufficiently in new C*.

We believe that a much simpler extension of C suffices to realize data parallelism. One needs to add a single new statement, namely a synchronous *forall,* plus perhaps its asynchronous form. For data structures, one needs to introduce true multidimensional arrays plus

pragmas that specify how to lay out the data. Such extensions have been implemented successfully in a compiler for the language Modula-2*, targeting the Connection Machine [5, 4]. A simple and consistent extension of Modula-2 avoids all the problems we have mentioned, without loss of efficiency. One might, however, call this work an unconfirmed experiment in language design and compiler construction. But this is exactly our point: More time is needed before we can standardize parallel programming languages.

At this time, design and compilation of parallel languages is in an experimental phase, as can be seen by the numerous proposals for such languages, but the reports are scant on experience with their implementation and use. However, knowledge in this area is increasing rapidly, so it would be imprudent to fix a poorly thought-out extension of a language as influential as C at this time.

Walter F. Tichy,
Michael Philippsen
University of Karlsruhe
School of Informatics
D-7500 Karlsruhe, F.R.G.
Phil Hatcher
University of New Hampshire
Department of Computer Science
Durham, NH 03824, U.S.A.

Response

Thinking Machines has proposed the C*® language as a data parallel language for programming massively parallel computers. We believe that C* provides a positive answer to many of the questions raised concerning ease of programming, ability of debugging, and efficiency of execution on these machines. We are proposing C* as one approach for an emerging parallel C language within the auspices of subcommittee X3J11.1 of ANSI technical committee X3J11 for programming language C. Thinking Machines, X3J11.1, and I, personally, would welcome comments and participation from all interested parties and are happy to address their concerns as the language evolves.

We have no intention to push standardization of the current C* language without possible changes, rather we are involving a community of users who are familiar with data parallel (SPMD) programming. The design of C* to date has been influenced by comments from users (both inside and outside of Thinking Machines), from computer architects, and from language designers. We intend to continue to encourage, to involve, and to be responsive to all comments about programming massively parallel computers from C* and our other languages.

Specifically regarding the letter by Tichy, Philippsen, and Hatcher, we have commented on all of their concerns in a paper (entitled "A Detailed Response to the C* Critique by Tichy, Philippsen, and Hatcher") which we are happy to send to any interested parties.

James L. Frankel
Thinking Machines Corporation
245 First Street
Cambridge, MA 02142-1264,
Frankel@Think.COM

REMARKS ON A DATA ENCRYPTION SCHEME OF YU AND YU

In the article "Superimposing Encrypted Data," (Feb. 1991 pp. 48–54), Yu and Yu consider a family of cryptosystems which enable the manipulation of encrypted data. The cryptosystems can be thought of as acting on time varying n-dimensional vectors $x(t) = (x_1(t), ...,x_n(t))$ and are defined using the basic building block,

$$x_i(t + 1) = f_i(x(t)) - x_i(t - 1) \quad (1)$$

or

$$x(t + 1) = f(x(t)) - x(t - 1) \quad (2)$$

where $f(x) = (f_1(x),...,f_n(x))$. (This is a slight generalization of Yu and Yu's scheme since they have each f_i varying only with the position i and not independently specified.) The initial values for the vectors $x(0)$ and $x(1)$ are defined by the plaintext to be encrypted, and the transformation is iterated a number of times T. The ciphertext consists of the two successive vectors $x(T)$ and $x(T + 1)$. It is not necessary for the function f to have any special properties, or even be invertible for decryption to be possible, so f can be an arbitrary key for the cryptosystem. However, in order to allow the encrypted data to be manipulated the function f is chosen to be linear in Yu and Yu's article.

The cipher has a striking resemblance to the class of Feistel ciphers which include the Data Encryption Standard [3], where the basic building block works on the two halves of the ciphertext block. If the original plaintext block is (M_0,M_1) then the transformation is defined by

$$M_{t+1} = f(k_t,M_t) + M_{t-1} \quad (3)$$

where the arithmetic is done bitwise modulo 2 and the k_t are subkeys of the full key. The parallel between (2) and (3) is clear. The major difference, however, is that the f function for DES is nonlinear. It is widely held that the use of linear functions in cryptography is very dangerous.

Yu and Yu consider an attack designed to find the function f in (1) by making use of its linearity and conclude that it is not possible. However, instead of directly solving for the linear function f, consider the mapping from plaintext to ciphertext. This takes two plaintext vectors and outputs two ciphertext vectors.

LEMMA 1: *If the map f: M −> M is linear then the map L: M x M −> M x M defined by L (x,y) = (f(x) − y),x) is also linear.*

If M is the space of n-dimensional plaintext vectors, then the cryptosystem of Yu and Yu is exactly the function L iterated T times, with *x* and *y* the initial vectors representing the plaintext. The following is easily proved.

LEMMA 2: *Any iteration of a linear function is itself linear.*

Therefore the entire encryption transformation is a linear function of the product space M x M. Since we know the encryption function is invertible its inverse is well defined and is also linear. This function can therefore be cryptanalysed by a known or chosen plaintext attack. Once the ciphertexts of 2n linearly independent message vectors are known, the encryption function is uniquely defined. By expressing the ciphertext as a linear combination of these known ciphertexts, the plaintext can be found. Note that the complexity of this attack is independent of the number of times the basic block is iterated and is feasible for very large message blocks. Using a very large message block will make the attack harder but will also affect the practicality of the cryptosystem for small messages and increase error propogation.

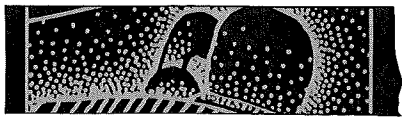
The use of a random "background" in place of one of the plaintext vectors, as suggested by Yu and Yu, does not affect this attack even if none of the random numbers used to encrypt the plaintexts were revealed. To see this suppose that the encryption function is a linear function L′ which satisfies

$$L'(m,r) = (x(T+1),x(T))$$

for any message vector m and random vector r. Then the linear inverse function D satisfies

$$D(x(T+1),x(T)) = (D_1(x(T+1),$$

$$x(T)),D_2(x(T+1),x(T))) = (m,r)$$



Here D_1 and D_2 are linear functions from M x M to M but only D_1 is of interest to the cryptanalyst and can be found as above by knowledge of 2n ciphertext and plaintext pairs.

As an example, the function used to encrypt foreign exchange in the paper of Yu and Yu can be broken by knowing the ciphertext of just 28 chosen plaintexts. For a known plaintext attack when the plaintexts cannot be chosen a larger number will be needed in order that a set of 28 linearly independent ones can be found. (However even with a limited set of linearly independent known plaintexts, all ciphertexts belonging to the linear subspace generated by these can be found.) The same is true of the other example given in the article. Unfortunately it seems that the very feature which allows manipulation of the ciphertext is doomed to allow cryptanalysis.

Colin Boyd
Communications Research Group,
Electrical Engineering Laboratories,
University of Manchester,
Manchester M13 9PL, UK
(Email: BOYD@uk.ac.man.ee.v1)

Response

We agree that Boyd's method can be used to break the encryption scheme when used in superimposing encrypted data. However, the scheme may be protected from his suggested attack by making the message block very long. For example, if we process 1,000 dollar-yen accounts at one time with each account consisting of 14 characters, then one has to solve 14,000 equations. It may take quite a lot of computing time to solve the equations. The block can be made even larger, depending on the application. It should be noted that even though the message contains 14,000 characters, the transformation function $f[\{x_i\}]$ can be confined to involve only the neighboring elements (the exact number may also be kept confidential) and thus can be computed with ease; the beginning and the ending of the whole message can be packed with some special characters and thus one does not have to know the rear part of the whole message before processing the front part of it.

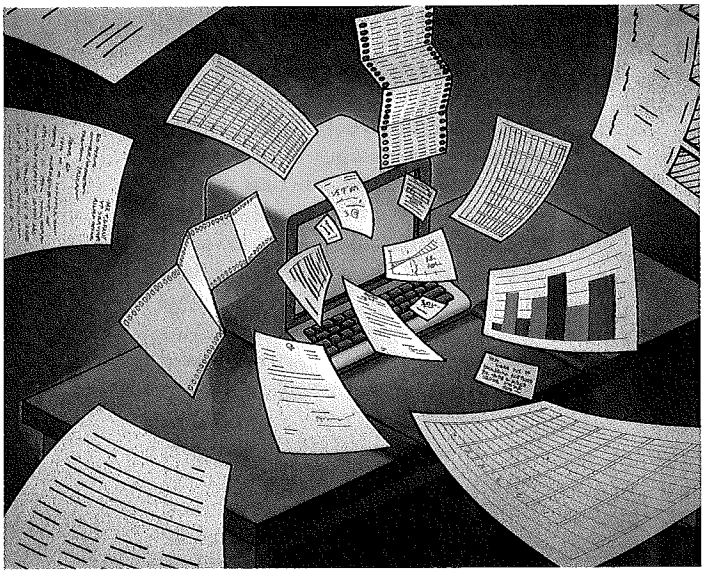
As we mentioned in our article, although homomorphic functions are susceptible to code breaking, we need them to superimpose encrypted data. The encryption scheme may be useful in small business applications. However, a lot more work needs to be done on this topic before it may be used safely. We are glad to see Boyd's comments.

Tong Lai Yu
Department of Computer Science
5500 University Pkwy.
San Bernadino, CA 92407

References

1. Blelloch, G.E. and Sabot, G.W. Compiling collection-oriented languages onto massively parallel computers. *J. Para. Distrib. Comput. 8,* 2 (Feb. 1990), 119–134.
2. Hillis, W.D., Steele, Jr., G.L. Data parallel algorithms. *Commun. ACM, 29,* 12 (Dec. 1986), 1170–1183.
3. National Bureau of Standards. Data Encryption Standard, Federal Information Standards publication 46 (Jan. 1977).
4. Philippsen, M. and Tichy, W.F. Compiling for massively parallel machines. In *Proceedings of the Workshop on Code Generation, Schloss Dagstuhl.* (May 20–24 1991), Springer Verlag, to appear.
5. Philippsen, M., Tichy, W.F. Modula-2* and its compilation. In *First International Conference of the Austrian Center for Parallel Computation,* (Salzburg, Austria, Sept. 30–Oct. 2), Springer Verlag, to appear.
6. Quinn, M.J. and Hatcher, P.J. Data-parallel programming on multicomputers. *IEEE Softw. 7,* 5 (Sept. 1990), 69–76.
7. Thinking Machines Corporation, Cambridge, Mass. *C* Language Reference Manual,* April 1991.

Collective Dynabases



ILLUSTRATIONS: BARTON STABLER

In my last column [15], I discussed portable computers as they are used for input to a person's dynabase, a dynamic database containing the notes, sketches, papers, and other documents he or she creates and gathers over a long period of time. The focus was on an individual's dynabase, but what about collective dynabases? For example, could the dynabases of everyone who worked for a company be combined, producing a corporate dynabase?

Much of the information an organization acquires or produces is already in machine-readable form. For example, text documents such as memos, letters, and reports are prepared on computers, and networks capture electronic mail. As portable computers and networks improve and proliferate, the percentage of information in machine-readable form will increase, making it still easier to create a dynabase as a byproduct of everyday work.[1]

Once information is in the dynabase, however, how will it be viewed and retrieved? To what extent will users have to add retrieval-enhancing information such as key words? To what extent will the system be able to generate such clues by analyzing a document or the context in which it was created? These are difficult questions, but if they sound insurmountable, do not despair—the system does not have to be perfect, just better than what we have today.

Researchers and commercial computing people are working on the challenge of collective dynabases. Let us look at examples of both, beginning with a research project on wide area information servers (WAIS).

Wide Area Information Servers

The WAIS project is headed by Brewster Kahle at Thinking Machines. Thinking Machines makes Connection Machines—highly parallel supercomputers that are well suited to free-text search [19]. In most text-retrieval systems, queries are limited to Boolean combinations of a few terms, but since text on a Connection Machine is fast, searches for documents which are similar to an entire document are practical. This technique is used in Dow Jones's DowQuest, a commercial system that uses a Connection Machine to scan more than 150,000 articles from 185 publications for relevance to online queries. While DowQuest is proprietary, the WAIS project is an attempt to open the technology.

WAIS is a client-server system, and as of this writing, there are 193 WAIS servers on the Internet, covering topics from poetry to television programs. Past issues of *Communications* are now available on a server.

Larry Press

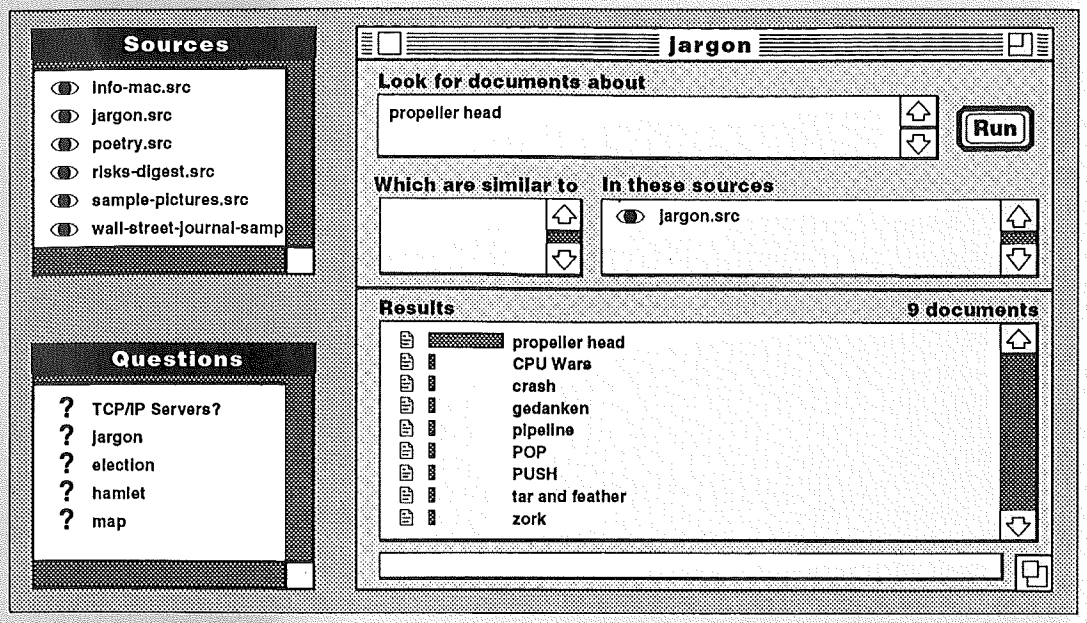
There is public domain client software for the Mac, DOS, Windows, NeXT and others. Since a standard protocol has been devised for queries and replies, any client can be used with any server. The protocol extends the NISO Z39.50 protocol developed for library catalog queries, and it is being used and developed by an active group of Internet-based researchers [12].[2]

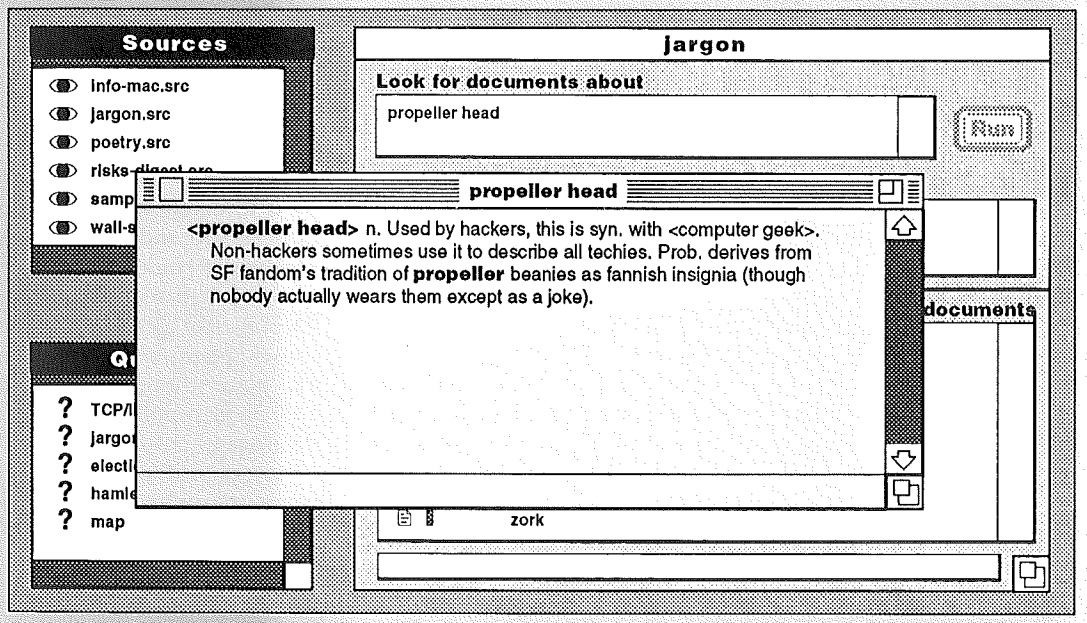
A WAIS session typically proceeds in two phases. First, you do a standard keyword search, identifying several documents. Then you search for documents similar to the most relevant ones from the first phase. Figure 1 shows a WAIS session using a Mac client. The session began with a query that sought documents about "propeller head" on the jargon.src server (Figure 1a). (Jargon.src contains the definitions found in [17], a delightful compendium of hacker jargon.) Nine documents were found, and their titles listed. Note that the length of the gray bar before the listed titles indicates the strength

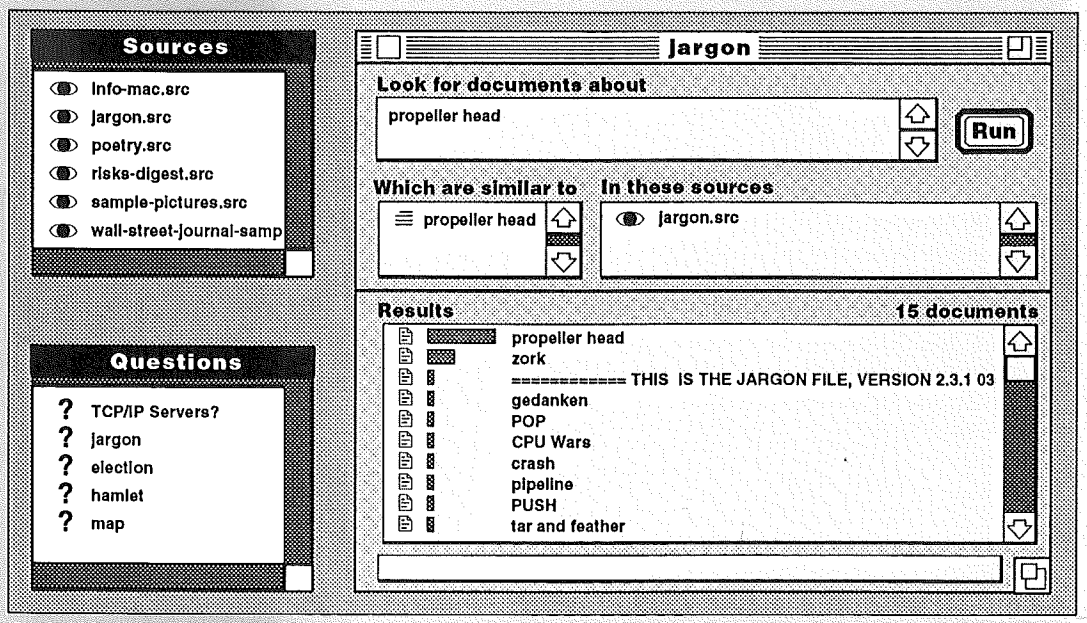
[1]Even if no extra effort is needed to enter information, individuals would have to give permission for information they create or gather to be published in the collective dynabase. Other issues affecting an individual's willingness to participate in such systems are discussed in [7, 15].

[2]Two other protocols, SFQL and CD-RDx, are also being developed for decoupling database queries. Structured Full-Text Query Language (SFQL) extends SQL, and it grew out of the need to query heterogeneous aircraft documentation distributed on CD-ROM [18]. It is being developed at the request of the Air Transport Association. CD-RDx is being developed at the request of the Information Handling Committee of the CIA (see [22]). Their goal is to enable government agencies to share data.

Figure 1. A WAIS session. This session involves two queries. First a keyword search for "propeller head" is run on the jargon.src database (1a). Nine documents are found, including one entitled "propeller head" (1b). The second search retrieves documents that are *similar to* the one called "propeller head" (1c).


A
Sources
Info-mac.src
Jargon.src
Poetry.src
risks-digest.src
sample-pictures.src
wall-street-journal-samp
Questions
TCP/IP Servers?
Jargon
election
hamlet
map
Jargon
Look for documents about
propeller head
Run
Which are similar to
In these sources
Jargon.src
Results
9 documents
propeller head
CPU Wars
crash
gedanken
pipeline
POP
PUSH
tar and feather
zork


B
Sources
Info-mac.src
Jargon.src
Poetry.src
Jargon
Look for documents about
propeller head
Run
propeller head
<propeller head> n. Used by hackers, this is syn. with <computer geek>. Non-hackers sometimes use it to describe all techies. Prob. derives from SF fandom's tradition of propeller beanies as fannish insignia (though nobody actually wears them except as a joke).
zork


C
Sources
Info-mac.src
Jargon.src
Poetry.src
risks-digest.src
sample-pictures.src
wall-street-journal-samp
Questions
TCP/IP Servers?
Jargon
election
hamlet
map
Jargon
Look for documents about
propeller head
Run
Which are similar to
propeller head
In these sources
Jargon.src
Results
15 documents
propeller head
zork
========== THIS IS THE JARGON FILE, VERSION 2.3.1 03
gedanken
POP
CPU Wars
crash
pipeline
PUSH
tar and feather

of a heuristic index of the degree of relevance. Any of the documents can be retrieved by clicking on it, and Figure 1b shows the document entitled "propeller head." Figure 1c shows the result of a second search, in which I asked it to find documents *similar to* the document called "propeller head." As you see, it found 15 such documents.

Although WAIS is experimental, it is easy to use, and the fact that the servers are spread over the Internet is well hidden. Automatic searches (e.g., a daily search of the *Wall Street Journal*) are also possible and bit maps and sound files are supported.

WAIS was also used as a corporate dynabase in a three-month experiment, run with the cooperation of Apple, Dow Jones, and Thinking Machines at the KPMG Peat Marwick consulting firm [9]. According to Robin Palmer, senior manager and WAIS project leader, WAIS integrated internal information such as word processing documents, letters, reports to clients, and training materials with external data like DowQuest and technical tax and accounting literature from the Financial Accounting Standards Board. Palmer is quite positive. In one instance they prepared a proposal for a new prospect on four hours notice. It included competitive and industry analysis and a discussion of related work by KPMG, and led to a contract on the first meeting. Palmer felt the internal data was most important in that it "reflected KPMG's internal knowledge." He warns, however, that three weeks were needed for initial data conversion, and that WAIS requires at least 56Kb/sec communication.

The Internet experiments and KPMG test are research prototypes, but they foreshadow what may be possible when networks are ubiquitous and Connection Machines are on desktops. Let us now look at some commercial developments.

Commercial Beginnings—Lotus Notes

Lotus is best known for its 1-2-3 spreadsheet, but many years ago it made a strategic commitment to groupware, and in 1989, began selling a system called Notes.[3] Notes is a step toward the corporate dynabase.

A company with only one location would see Notes as a LAN-based conferencing system with a graphical user interface (see Figure 2). The LAN would have a dedicated Notes server with many databases.[4] Notes databases contain documents of one or many formats, and field data types include dates, numbers, free text, and rich text (text with varying fonts and bit-mapped graphics and check-box categories).

Users at client stations post documents to databases by filling in screen forms, and if they have permission, view and edit them. The user interface conforms to IBM's Common User Access Standard, and users can search and view a database in many ways, for example selecting only certain records and fields to be displayed or changing their order. The database designer would provide some default views, and users are free to create custom views. Totals, averages and other statistics can be computed, and more complex operations may be programmed using the Notes API, which gives access to databases.

If this were all it did, Notes would be quite useful, but there is another key aspect. Notes is not restricted to single locations. It was designed for distributed organizations that do not have a persistent data link between sites [10]. If an organization had two or more locations, each would have a LAN with a Notes server. At time intervals set by the system administrator, connections would be established between sites, and the servers would exchange all new data, leaving the databases at both sites identical. Notes also supports users with portable or stand-alone computers who dial into a server. This database replication is similar to periodic updates between network directory servers, and while it would not work for transaction processing and other applications requiring immediate synchronization, it is sufficient for a corporate dynabase.

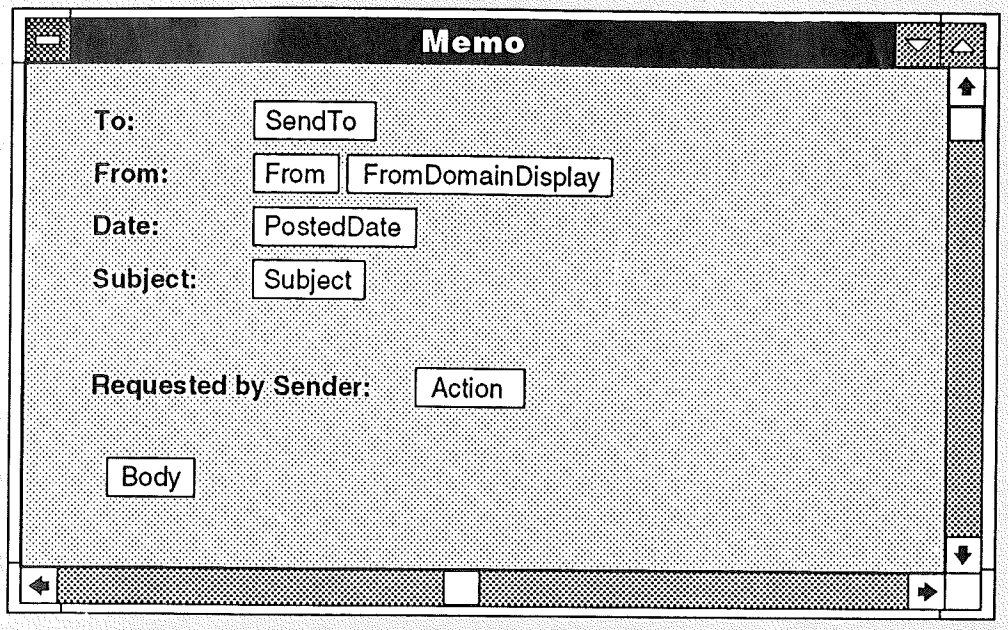
Notes is sold to large organizations (the minimum installation is 200 users at a price of $62,500), and there is now considerable experience with it. For example, Marshak reports that Notes is being used to create an environment of information and processes at Manufacturers Hanover Trust, a large New York bank [13]. The bank's goal is to "get the information stored in people's heads into a form and place where others can use it," and their applications include a database of customer and potential customer profiles and a database of communications with customers. Marshak also reports on the Price Waterhouse accounting firm. Of its 12,000 employees, 6,000 are Notes

Figure 2. Lotus Notes. This database holds electronic mail messages sent and received. When a database is defined, one or more forms are created (2a). For example, a mail database might have records for general messages (shown here), replies, and phone messages. The designer specifies form layout and access control, field data types, default values, formulas for validity checking and optional value computation, and other field and form properties. Views are also defined. For example, (2b) shows messages sorted by user-assigned category, but others like view by name or data could also be defined. In 2c, the user has displayed one record. Note that date and category check-boxes are built-in types, and the rich-text field with the message body contains a bit-mapped drawing.

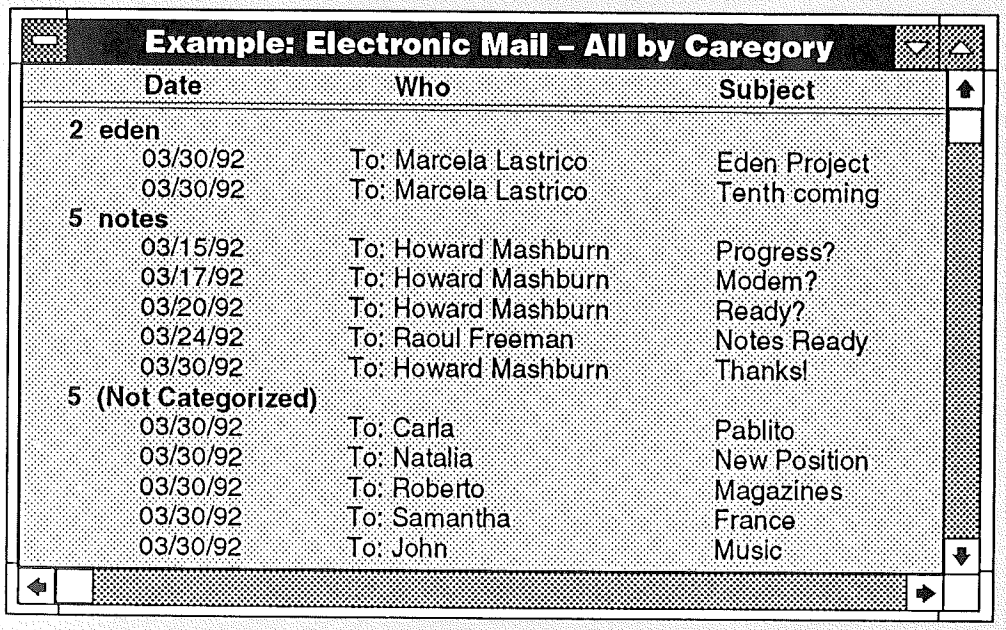
[3] It is interesting to speculate on how companies make such decisions. Perhaps proximity to a nearby research project plays a role. Lotus was in Cambridge, and hired groupware researcher Irene Greif from MIT, and Apple was near Xerox PARC, where they got ideas and people for the Lisa and Mac. Notes was developed for Lotus by Ray Ozzie who was hired for another product, but had wanted to build a conferencing system since his college days. How did MicroSoft decide to move into CD-ROM and multimedia?

[4] Today Lotus has only OS/2-based servers and Windows and OS/2-based clients, but servers for Unix, Netware and NT and Mac and Unix clients are under development.

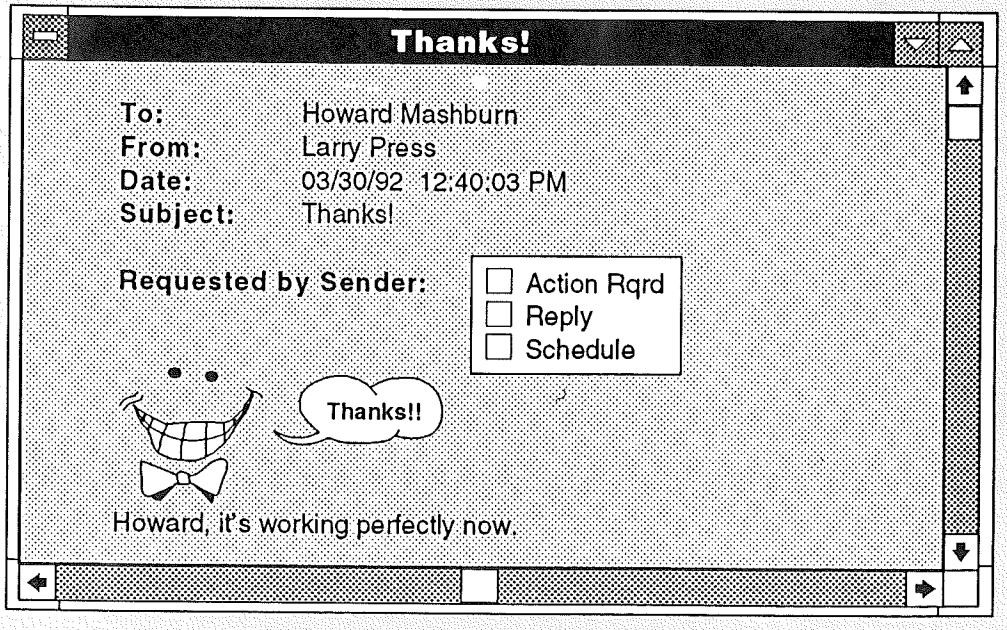
A


Memo
To:
SendTo
From:
From
FromDomainDisplay
Date:
PostedDate
Subject:
Subject
Requested by Sender:
Action
Body

B


Example: Electronic Mail – All by Caregory
Date
Who
Subject
2 eden
03/30/92 To: Marcela Lastrico Eden Project
03/30/92 To: Marcela Lastrico Tenth coming
5 notes
03/15/92 To: Howard Mashburn Progress?
03/17/92 To: Howard Mashburn Modem?
03/20/92 To: Howard Mashburn Ready?
03/24/92 To: Raoul Freeman Notes Ready
03/30/92 To: Howard Mashburn Thanks!
5 (Not Categorized)
03/30/92 To: Carla Pablito
03/30/92 To: Natalia New Position
03/30/92 To: Roberto Magazines
03/30/92 To: Samantha France
03/30/92 To: John Music

C


Thanks!
To: Howard Mashburn
From: Larry Press
Date: 03/30/92 12:40:03 PM
Subject: Thanks!
Requested by Sender:
Action Rqrd
Reply
Schedule
Thanks!!
Howard, it's working perfectly now.

users. While only one-third of their top management had used PCs before Notes was installed, all are now active Notes users. Notes is seen as a strategic tool, and the Price Waterhouse chairperson characterized one of the firm's major challenges as the need to collect, manage, and access "the expertise of our people."

We shape our tools and they shape us. Information processing tools and organizations co-evolve, and Notes affects and is affected by corporate culture. Ed McDonald, manager of the Information Processing Division at Texaco Oil, states that they have 1,500 Notes users. Before they began using Notes, Texaco had decided to alter the corporate shape and culture, moving from a rigid hierarchy to a flatter organization with freer lines of communication and more information sharing. McDonald feels Notes would not have been as well received without this move, and that, in turn, it has facilitated the shift. As an example, before Notes, his subordinates sent him weekly activity reports. Now they post the reports on Notes, and everyone can see what the others are doing (and how hard they are working).

Texaco also uses Notes for distributed synchronous meetings. Each month, 20 people from eight U.S. locations meet, using an audio link and Notes. Notes is used to distribute the agenda before the meeting and for note taking and brainstorming during the meeting (each user logs into the same server). The major Notes controversy is a debate over the use of company resources to support an informal "coffee bar" conference where people shoot the breeze as when on coffee breaks.

With time, we will gather systematic data on the impact of such systems. Brad Jackson of Texaco is collaborating with researchers at the University of Minnesota in a two-year study of the impact of group decision support tools including Notes. Wanda Orlikowski and her colleagues at MIT are studying the application and impact of Notes in several organizations. In the meantime, customers are installing Notes (e.g., Arthur Anderson has purchased 20,000 licenses and committed to 60,000), and it is a strategic product which Lotus is extending rapidly. Lotus has also formed key business alliances. For example, IBM markets Notes, Kodak is working with Lotus on document-imaging, Verity is adding their text-retrieval capability, and Notes is being implemented as a Novell Netware Loadable Module, which will eliminate the need for an OS/2 server (but may cut speed).

While Notes gives Lotus a lead, it is only a first step. People will not use a corporate dynabase if doing so is difficult. For a start, seams between the dynabase and other manual and software tools must be closed [8]. Lotus, MicroSoft and others are mail-enabling their applications. Spreadsheet and word processing users will soon be able to *mail* a document or portion of a document as easily as they now *print* or *save* it. Instead of importing files, users will be able to use their word processor while working in Notes. More integration is needed. A successful dynabase requires natural interfaces to OCR systems, portable computers, pens, faxs, telephones, and wide-area networks. Stitching seams will foster building the dynabase as a byproduct of other work.

Standards are also needed. Today Notes is self-contained, but it is being rewritten to run on top of VIM, Lotus's Vendor Independent Messaging interface. IBM, Apple, Novell, Borland, and others are committed to VIM. MicroSoft is not because they have their Messaging Application Program Interface (MAPI). Such standards will not only allow software vendors to mail-enable current applications, they will use them for new applications such as scheduling and workflow. In-house programmers will also use them for custom applications.

While Lotus is ahead with Notes, MicroSoft is ahead with MAPI. They have shipped a software developer's kit, and two-thirds of the over 300 people in their Workgroup Applications Business Unit are working on system software. MAPI and VIM overlap in the services they support, and it would be premature to guess which will (or should) prevail. Technical people from both camps are in contact. It took roughly 50 years to establish a standard, unified telephone system in the U.S. [14]. In the long run, communication services will be part of operating systems, and all applications will use them. As with the telephone, the standards will be decided by the market place and corporate power, not just technical merit.

While formidable, solutions to the problems of bridging seams and developing standards are understood. Dynabase retrieval and organizational problems are more open-ended. The ideal dynabase would have a simple interface. The user would present it with an arbitrary document, and either ask that it be filed away or that similar documents be retrieved. That vision calls to mind vague prospects for natural-language processing for classifying and indexing text or recognized speech documents and pattern recognition for graphics. While we are far from such capability, an interim dynabase can use heuristics and file inversion to index text (as WAIS does). It can relate items by the date and context

In the long run, communication services will be part of operating systems, and all applications will use them.

in which they were created (for example the words on a page surrounding a sketch or the project you were working on when you made it), and allow the user to assign key words, degrees of importance, and other explicit clues.

Beyond the Corporate Dynabase

Information systems cross corporate boundaries. For example, manufacturers send email to suppliers and customers, and Electronic Data Interchange (EDI) is used for intercompany transactions. The infrastructure being built to support such activity will be available for interorganizational dynabases.

Patricia Seybold's Office Computing Group has created an interorganization dynabase. Seybold has traditionally done consulting and published newsletters and research reports. Last year the group began publishing a Notes database containing the information in the printed versions of their four monthly newsletters plus special bulletins and analysis written for the Notes group. This information gets to clients immediately, and it can be searched and viewed in any way the reader desires. Note that this is not just electronic distribution of a newsletter (as if it were being faxed); it is a two-way link with clients who make comments and guide the direction of further research. Seybold has a sweeping vision, and expects most of her clients to eventually switch from printed to electronic reports within two to three years.

Ken Laws is not phasing out printed newsletters. His year-old "Computists Communique" was electronic from the start, and is available on the Internet.[5] Each issue covers news of the domestic and international computer and research industries with topics such as job opportunities, calls for papers, and conference announcements. Another section has pointers to newsletters, databases, software and other Computists' tools. These are followed by commentary and an in-depth essay on a related topic. There is something noteworthy (worth saving in your dynabase) in every issue. Similar to Seybold, Laws is creating a community. Issues are peppered with comments from and pointers to readers, and past issues are retrievable.

My last column discussed individual dynabases, fed from portable computers. This one has moved to organization and interorganization dynabases, but speculative writers consider the entire planet. The French paleontologist, theologian and philosopher Pierre Teilhard de Chardin [20] held that the matter-energy comprising the universe is constantly evolving in the direction of increased complexity, leading to the formation of the earth, the "geosphere," and life, the "biosphere." In 1925 he coined the term "noosphere"—an evolving network of human culture, connection, knowledge and interdependence. Zoologist Richard Dawkins [2] discusses "memes"—replicable ideas that live in people's minds. Dawkins holds that memes are not mere metaphors, but are literally realized in the physical structure of the nervous system of the individuals in which they reside. If the earth is an evolving organism, perhaps the Internet is a step in the evolution of the Gaian nervous system, and you and I and WAIS servers are neurons.

[5]Laws began electronic publishing as the moderator of the Internet AI list.

References:

1. Brin, D. *Earth.* Bantam Books, New York, 1990.
2. Dawkins, R. *The Selfish Gene.* Oxford University, Oxford, 1976.
3. Engelbart, D.C. Knowledge-domain interoperability and an open hyperdocument system. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work,* (Los Angeles, CA, Oct. 1990), 143–156.
4. Engelbart, D.C. and English, W.K. A research center for augmenting human intellect. In *Proceedings of the AFIPS Fall Joint Computer Conference* (San Francisco, Dec. 1968), pp. 395–410.
5. Erickson, T. and Salomon, G. Designing a desktop information system: Observations and issues. *CHI '91 Conference Proceedings,* Addison-Wesley, Reading, Mass., 1991, pp. 49–54.
6. Gorry, G.A., Long, K.B., Burger, A.M., Jung, C.P., and Meyer, B.D. The virtual notebook system™: An architecture for collaborative work. *J. Org. Comput. 1,* 3 (1991), 233–250.
7. Grudin, J. Why CSCW applications fail: Problems in the design and evaluations of organizational interfaces. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work* (Portland, Oreg., Sept. 1988).
8. Ishi, H. and Miyake, N. Toward an open, shared workspace. *Commun. ACM* (Dec. 1991), 37–50.
9. Kahle, B. and Medlar, A. An information system for corporate users: Wide area information servers. FTP: pub/wais/doc/wais-corp.txt@think.com, Apr. 8, 1991.
10. Kawell, L., Beckhardt, S., Halvorsen, T., Ozzie, R. and Greif, I. Replicated document management in a group communication system, in Marca D. and Boch, G. *Groupware: Software for Computer-Supported Cooperative Work.* IEEE Press, to be published.
11. Lovelock, J.E. *Gaia.* Oxford University, Oxford, 1979.
12. Lynch, C.A. The Z39.50 Information retrieval protocol: An overview and status report. *ACM SIGCOMM Comput. Commun. Rev. 21,* 1 (Jan. 1991), 58–70.
13. Marshak, D.S. Lotus notes, a platform for group information management applications. Special Research Report, Patricia Seybold's Office Computing Group, Boston, Mass., June, 1991.
14. Pierce, J.R. *Signals, The Telephone and Beyond.* Freeman, San Francisco, Calif., 1981.
15. Press, L. Dynabook revisited. *Commun. ACM* (Mar. 1992).
16. Press, L. Systems for finding people. *J. Org. Comput.,* to be published.
17. Raymond, E. *The New Hacker's Dictionary.* MIT Press, Cambridge, Mass., 1991.
18. Shapiro, N.R. Diamantopoulos, E.

and Cotton, P. CD-ROM Disc Interchangeability Standards: Beyond ISO 9660 with Structured Full-Text Query Language (SFQL), ATA/AIA 89-9C Monograph, April 1991.

19. Stanfill, C. and Kahle, B. Parallel Free-Text search on the Connection Machine system. *Commun. ACM 29,* 12 (Dec. 1986), 1229–1239.

20. Teilhard de Chardin, P. *The Phenomenon of Man.* Harper and Row, N.Y., 1955.

21. WAIS interface protocol, prototype functional specification. Apr. 1990. Anonymous FTP: /pub/wais/doc/protspec.txt@think.com.

22. Standard for the exchange of digital information on CD-ROM. DCI Information Handling Committee, Intelligence Community Staff, Wash., D.C., 20505, Version 3.1, May, 1991.

Pointers

• For more information on WAIS, contact info@think.com on the Internet. Papers about WAIS can be retrieved by anonymous ftp from /pub/wais/doc@think.com. To retrieve Communications of the ACM articles, log on and select the database called "cacm". If you do not have Internet access, Thinking Machines Corp. is at 245 First Street, Cambridge, MA 02142-1264; tel: (617) 234-1000; fax: (617) 234-4444. For a discussion of interface issues, WAIS, and DowQuest, see [4].

• The 1992 revision of the Hacker's Dictionary is 100% fun. It is the perfect programmer's aptitude test. If you feel like putting it down in less than one hour after picking it up, look for another profession. (This suggestion is only half joking —people with an aptitude for a symbol-manipulating profession such as programming might have similar taste in humor and wordplay).

• Patricia Seybold's Office Computing Group does consulting and publishes material about groupware, object-oriented technology, and a variety of other topics. This group covers technology, the industry, and impact on organizations. 148 State Street, 7th Floor, Boston, MA 02109; tel: (800) 826-2424; fax: (617) 742-1028; pseybold@mcimail.com.

• Science fiction fans who would like to pick up where this column ended, will enjoy David Brin's book *Earth 1,* 1990.

• Ken Laws, 4064 Sutherland Drive, Palo Alto, CA 94303; tel: (415) 493-7390; email: laws@ai.sri.com.

• Any discussion of corporate memory needs a pointer to Doug Engelbart. Engelbart inspired a generation of researchers when he demonstrated his NLS system at the 1968 Fall Joint Computer Conference. For a description of his early work see [4], and for his current views on interorganization dynabases, see [3]. Much of the 1968 demonstration is shown on the video tape of ACM's Conference on the History of Personal Workstations. Doug Engelbart, The Bootstrap Institute, 6505 Kaiser Drive, Fremont, CA 94555; tel: (510) 713-3550; fax: (510) 793-2362; email: engelbart@bootstrap.stanford.edu.

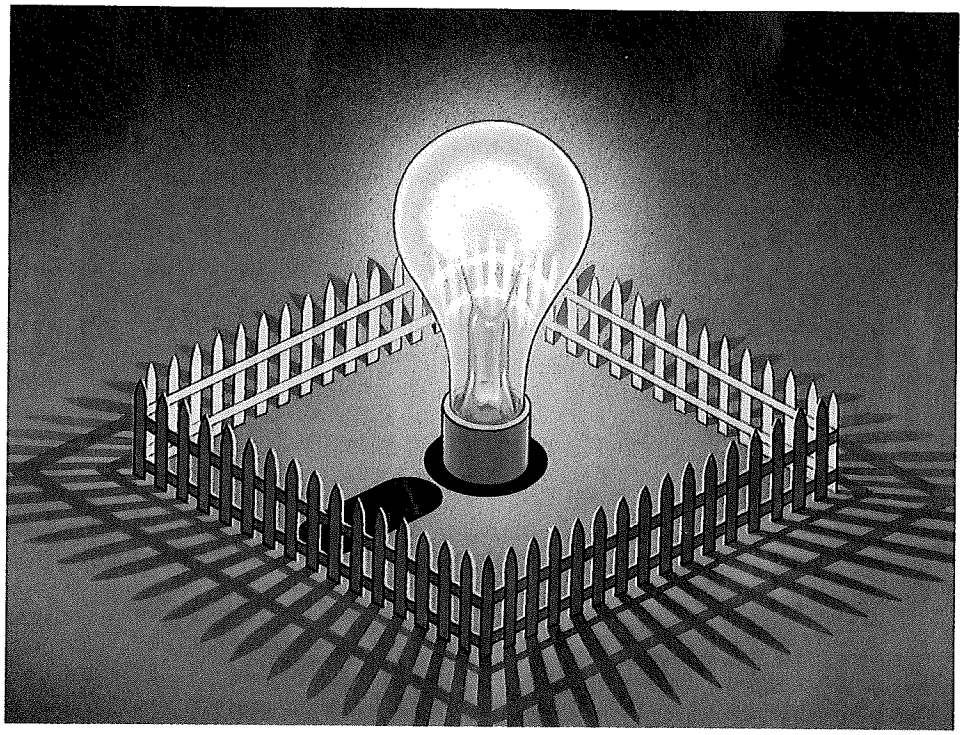
• Of course Notes and WAIS are not the only systems working toward organizational memory. Another noteworthy effort is the Virtual Notebook System™, a networked, multimedia system for recording lab notes and other research-related information [6]. Shared electronic notebooks contain text, image, audio or video entries, and notebook objects can be linked to other objects or to executable programs. The system is being developed at Baylor College of Medicine, and tested there and at several other sites. A commercial release is available from The Fore-Front Group, 1709 Dryden, Suite 901, Houston, TX 77030; tel: (713) 798-6116; fax: (713) 798-3729; email: klong@bcm.tmc.edu.

• Corporate and intercorporate dynabases are studied in schools of business. You might be interested in the annual Conference on Organizational Computing, Coordination and Collaboration, which brings together business school faculty and industrial practitioners in a workshop-like setting. The papers presented appear in special issues of the *Journal of Organizational Computing,* Ablex, Norwood, N.J. For information contact Andrew Whinston, abw@emx.utexas.edu.

Larry Press welcomes questions and comments from readers. His address, phone number and email are: 10726 Esther Avenue, Los Angeles, CA 90064; (310) 475-6515; lpress@venera.isi.edu

Larry Press is a professor of computer information systems at California State University at Dominguez Hills.

Developments on the Intellectual Property Front



ILLUSTRATIONS: BARTON STABLER

The rift between what computing professionals think the law of intellectual property rights in computer programs ought to be and what intellectual property professionals (mainly lawyers) think it ought to be is growing wider every day. At the moment, it appears that the intellectual property professionals are outmaneuvering the computing professionals by working toward establishing their vision of the proper rules on software intellectual property rights as "the law" before the computing professionals even know that the rules that will govern their conduct are being decided.

While there are unquestionably pros and cons to the software patent and other intellectual property controversies, the unfortunate fact of current U.S. policy on intellectual property rights for such an important product as computer programs is that the policymaking seems largely to be occurring either behind closed doors or in courtrooms across the country in cases in which the court papers are filed under seal. This effectively precludes those whose work will be substantially affected by the resolution of these controversies from having any meaningful input into the process of shaping the law in a manner that would make sense to them. Exclusion of computing professionals from the policymaking process also means the opportunity to persuade them of the merits of proposals eventually adopted has been lost. This, in turn, may have serious consequences for the enforceability of the proposals if they become the law.

This column will report on this rift by bringing readers up to date on some national and international developments in the intellectual property rights arena and by reporting the results of a survey on intellectual property rights conducted in August 1991 at the SIGGRAPH conference in Las Vegas. The SIGGRAPH survey results are much the same as the CHI '89 survey results reported in the May 1990 "Legally Speaking" column (pp. 483-487). Both surveys show strong support for copyright protection for source and object code, but little support for copyright or patent protection for most aspects of user interfaces and internal structural features of computer programs. If anything, the SIGGRAPH survey results show even stronger opposition to copyright protection for look and feel than did the CHI '89 survey, as well as stronger opposition to patent protection for algorithms.

Further evidence of significant opposition to patent protection for computer program-related inventions can also be found in a large number of letters sent by computing professionals in response to last summer's call for public comment by a U.S. Advisory Commission on

Pamela Samuelson, Michel Denber, and Robert J. Glushko

Patent Reform that was ostensibly created to address questions about patent protection for software innovations, among other issues. The Commission's recently released draft report dismisses concerns raised by software patent opponents, and urges, if anything, broadening the role of patents for software innovations. That the Commission should be preparing to make these recommendations is not surprising to those who know the composition of the subcommittee in charge of the computer program-related invention issues. This aspect of the Commission's work seems to be a thinly disguised effort to prevent a more democratic public debate on software patent issue in which the views of computing professionals could be considered.

Other events happening outside the realm of public debate include the recent release of a draft agreement on intellectual property rights being considered for inclusion as an addendum to the General Agreement on Tariffs and Trade (GATT). Although the draft does not directly say so, one of its provisions can be read as requiring member nations to provide patent protection for software innovations. This aspect of the GATT-related draft agreement would seem to implement another recommendation of the U.S. Advisory Commission on Patent Reform draft report which urges the U.S. to strongly encourage other countries to broaden patent protection for program-related inventions.

The SIGGRAPH Survey

At SIGGRAPH '91, a panel on intellectual property rights organized by Xerox researcher Michel Denber featured four speakers: John Perry Barlow, songwriter and a co-founder of the Electronic Frontier Foundation; Isaac Kerlow, computer artist and chair of the Computer Graphics Department at the Pratt Institute; Peter Deutsch, chief scientist at ParcPlace Systems, and Pamela Samuelson, professor of law specializing in software intellectual property law. (Denber tried to persuade a number of representatives from firms involved in some of the well-publicized lawsuits to participate on this SIGGRAPH panel to explain why their firms' positions in the lawsuits will be good for the field, but no firm was willing to have their people comment until the lawsuits are decided.)

After the panelists spoke and answered questions, the audience was asked to respond to a survey nearly identical to the one on intellectual property rights conducted at CHI '89. There were 345 respondents to the SIGGRAPH intellectual property rights survey. As with the CHI '89 survey (which had 667 respondents), the SIGGRAPH survey was filled out by people who mainly worked for firms that develop software for commercial purposes (only 1-in-5 of the respondents to these surveys worked for universities). As one might expect, a higher proportion of the CHI '89 survey respondents identified themselves as user interface designers and human factors engineers than did the SIGGRAPH survey respondents, and more of the SIGGRAPH respondents identified themselves as computer artists and programmers, but otherwise the respondent demographics were quite similar (including the 15-16% who reported being managers on both surveys). This means there are now approximately 1,000 computing professionals who have made their views known on the major intellectual property controversies of the day.

Overview of Findings on Various Aspects of Software Protection

There were three aspects of programs that enjoyed significant support for intellectual property protection among the SIGGRAPH survey respondents. Like the CHI '89 respondents before them, SIGGRAPHians overwhelmingly supported copyright protection for the source code of computer programs. Although a strong majority also supported copyright protection for object code (as had the CHI survey respondents), support for copyright protection for object code was nonetheless lower among both SIGGRAPH and CHI respondents than was the support for copyright for source code. The other aspect of software that enjoyed strong support for copyright protection from SIGGRAPH respondents was computer-generated images (a subject about which no inquiry was made on the CHI survey). Table 1 reflects the results of this part of the SIGGRAPH survey.

Although the SIGGRAPH survey results on user interface issues were quite similar to those from CHI '89, there were some differences in their views on protection for internal design elements of programs. A slight majority (52%) of CHI '89 survey respondents had supported copyright or patent protection of pseudocode whereas only 39% of SIGGRAPH respondents favored protection for it. There was also somewhat less support among the SIGGRAPH respondents for copyright or patent protection for modular design. Some 40% of CHI respondents had favored protection for modular design, but only 28% of the SIGGRAPH respondents favored such protection.

The most striking contrast between the SIGGRAPH and the earlier CHI survey results was the dramatically lower support for patent protection for algorithms among SIGGRAPHians. A total of 39% of the CHI survey respondents had favored patent protection for algorithms, and another 8% favored copyright protection for them. This was less than a majority opinion, but among the SIGGRAPH respondents, only 13% favored patent protection for algorithms (with another 9% favoring copyright protection for them). To put it a slightly different way, nearly 4 out of 5 of the SIGGRAPH respondents were against patent or copy-

right protection for algorithms, whereas the CHI respondents were almost evenly split on the issue.

A number of respondents commented on software patent issues in the blank space at the end of the SIGGRAPH survey. One person expressed the view that patents could provide significant protection for smaller software developers against "theft" of their ideas by larger firms. Others commented that patents on truly inventive ideas would be OK, but thought that too many patents had been granted to trivial things. Still others objected to the duration of patents, suggesting that five years might be a more appropriate duration than 17 years. Another comment suggested that those who independently developed an idea should not be precluded from using it, and objected to patents because of the completely exclusionary character of this form of intellectual property protection.

Stronger Opposition to Look and Feel

Opposition to copyright protection for the look and feel of computer programs was also stronger among the SIGGRAPH respondents than among the CHI '89 respondents. More than three-quarters of the CHI respondents had expressed opposition to protection for the look and feel of computer programs. Of the SIGGRAPH respondents, however, 94% were opposed to look and feel protection.

The SIGGRAPH survey questionnaire asked the same two questions on predicted effect on the respondent's own work and on the industry if the currently pending copyright lawsuits established strong protection for the look and feel of user interfaces. As with the SIGCHI survey, a 5-point scale was used to gauge the respondents' predictions. (see Table 2)

The average predicted effect of strong look and feel protection on one's own work from the SIGGRAPH survey was 2.12; the average for the CHI '89 survey was 2.049. A larger percentage of the SIGGRAPH than CHI respondents (28% vs. 19%) did not expect an effect on their own work. Even so, 68% of the SIGGRAPH respondents anticipated a negative effect on their own work, and only 4% thought that strong protection would have a positive effect on their work.

As with the CHI survey, more SIGGRAPH respondents predicted negative effects for the industry than for their own work if the look and feel lawsuits established strong protection for user interfaces. The average predicted effect on the industry score on the same 5-point scale was 1.58 for the SIGGRAPH survey; it had been 1.646 in response to the CHI survey. Only 4% of the SIGGRAPH respondents predicted a positive impact on the industry if the look and feel lawsuits were successful, whereas 10% of the CHI respondents had anticipated a positive impact on the industry from success by plaintiffs in these suits. Only 7% of SIGGRAPH respondents expected no effect on the industry (quite similar to the 4% response from the CHI survey) if look and feel protection became the law. The identical percentage of SIGGRAPH and SIGCHI respondents—namely 57%—expected strong negative consequences to the industry from strong look and feel protection, as Table 2 demonstrates.

Table 1.
Support for Protection by Copyright or Patent*

	Copyright	Patent	Both	Neither	N
For:					
source code	86%	2%	3%	8%	318
object code	65%	2%	3%	27%	293
pseudocode	37%	1%	1%	61%	278
module design	18%	9%	1%	72%	269
algorithms	9%	12%	1%	79%	303
UI commands	6%	1%	0	92%	294
icons	43%	0	1%	56%	307
UI layout	19%	1%	1%	79%	302
UI sequences	9%	1%	0	90%	295
look and feel	5%	0	0	94%	312
UI functionality	5%	4%	0	91%	300
comp images	81%	1%	0	18%	316

*of those expressing an opinion

Table 2.
Predicted Effect of Look and Feel Protection

On own work:	1	2	3	4	5
SIGGRAPH	28%	40%	28%	2%	2%
SIGCHI	35%	36%	19%	7%	2%
On the industry:					
SIGGRAPH	57%	32%	7%	3%	1%
SIGCHI	57%	29%	4%	7%	3%

1 = strongly negative, 3 = no effect, 5 = strongly positive

Similar Results Concerning Other UI Features

Apart from the stronger opposition to look and feel protection, the SIGGRAPH survey yielded quite similar results to the CHI '89 survey concerning other aspects of user interfaces. A total of 92% of SIGGRAPH respondents opposed protection of user interface commands, as had 88% of the CHI respondents. In addition, 91% of SIGGRAPH respondents opposed patent or copyright protection for user interface functionality, as had 83% of CHI respondents. There

was somewhat less support among the SIGGRAPH than CHI respondents for protection of user interface screen layouts (79% opposition among SIGGRAPH and 69% among CHI respondents) and for user interface screen sequences (90% opposition among SIGGRAPH and 79% among SIGCHI respondents). Icons, however, were thought deserving of protection by almost equal percentages of SIGGRAPH (44%) and SIGCHI (43%) respondents.

One additional question asked on the SIGGRAPH survey that was not asked on the CHI survey was whether copyright or patent protection should be available for computer-generated images. Of the SIGGRAPH respondents, 81% favored copyright protection for such images. (Another 1% thought patents should be available for them). The difference between the computer-generated images and icon responses was somewhat surprising given that icons are just little images. Michel Denber thought the difference in responses to these two questions might be due to a perception by people in the SIGGRAPH community that there are comparatively few effective ways of iconically representing particular functions, but an infinite number of interesting computer-generated images. Because of this, people may not want to be forbidden from using an effective icon, but may not object to protection of computer art. Artistic expression is based on a much less constrained intellectual space, where the existence of one image does not preclude the creation of others on a similar theme.

Other Survey Findings

As with the CHI '89 survey, the SIGGRAPH survey asked respondents how constrained they currently felt about the uses they could make of research and design innovations they saw at SIGGRAPH after the conference. The results were quite similar to those obtained in response to a nearly identical question on the CHI '89 survey. Some 31% of the SIGGRAPH respondents (as compared with 31% of the CHI respondents) felt no restriction; that is, they felt they could freely use anything they learned about or saw at the conference.

As compared with 49% of CHI respondents, 54% of SIGGRAPH respondents felt some restriction; that is, they felt that they could not copy exactly, but could reimplement or reengineer any interesting designs they saw at the conference. Also, 14% of the SIGGRAPH respondents (as compared with 19% of CHI respondents) felt significant restrictions; they could copy only general concepts or ideas at the research stage. One percent of both groups felt totally restricted; that is, once they saw something at SIGGRAPH, they felt they could not copy it into a work of their own.

Those who attended the SIGGRAPH intellectual property panel felt reasonably familiar with these kinds of legal issues. A total of 17% indicated they felt moderately familiar with the issues before attending the panel. Another 18% felt highly familiar with them. Only 11% indicated they had not been familiar with the issues before attending the panel session. Only one-quarter of the SIGGRAPH respondents noted that attending the panel session had changed their views on the legal issues. (In contrast, half of the respondents had changed their minds on the issues after hearing the CHI debate.) Of those whose minds were changed as a result of attending the SIGGRAPH session on intellectual property rights, only 1 in 8 thought that protection should be stronger after hearing the issues discussed. (The CHI survey showed a respondent shift in the same direction, although by a different margin. Only 1 in 11 changed his or her mind to thinking that protection should be stronger after hearing the CHI debate.)

As with the CHI '89 survey, there was majority support for the idea that the SIG should use the results of the survey to take an official position on the legal issues. Some of those who opposed this idea did so because the respondent group was not a representative sample of the SIGGRAPH membership. Several SIGGRAPH respondents expressed the view at the end of the survey that ACM and their SIG should get more actively involved in the legal issues, not only by educating its membership about them, but also by lobbying Congress about changes in the law or taking a stand in some lawsuits. (In fact, ACM's Executive Committee has approved a proposal for beginning an investigation of software intellectual property issues. However, efforts have yet to get underway.)

Developments on the Patent Front

About two years ago, after some National Research Council workshops aired conflicting views on software intellectual property issues, a Congressional hearing was held on software intellectual property issues. At this hearing, software developers Mitch Kapor and Dan Bricklin, among others, expressed a number of concerns about patent protection for software innovations. Some of the concerns pertained to problems with how the U.S. Patent and Trademark Office (PTO) was implementing its policy on computer program-related inventions (e.g., problems arising from the PTO's ignorance of the prior art and too low a standard as to what software innovations were inventive enough to be patented). Some concerns were more fundamental in nature (e.g., whether patent protection for software innovations might significantly raise the barriers to entry for the software industry, especially worrisome because small software firms have been at the forefront of innovation in this industry).

At about the same time, the U.S. began to consider proposals to

The most striking survey contrast was the dramatically lower support for algorithm patent protection among SIGGRAPHians.

change its patent law to make it more like the patent laws of other industrialized nations. To address questions that have arisen concerning patent protection for computer program-related inventions (including those raised at the Congressional hearing) and to consider the patent harmonization proposals and some other issues, the U.S. Department of Commerce established an Advisory Commission on Patent Law Reform.

Although one important set of issues to be addressed by the Commission concerned software patents, no effort was made to find a prominent computing professional who had no stated position on the issues to serve on the Commission. The person appointed to serve as chair of the Commissions' working group on the computer program-related inventions was Howard Figueroa, an IBM executive who had publicly spoken in favor of patent protection for computer program innovations before his appointment to the Commission. (Interestingly, 20 years ago IBM was one of a number of computer firms who submitted an amicus brief to the U.S. Supreme Court in the *Gottschalk vs. Benson* case arguing against patent protection for algorithms and other program-related inventions because of their mathematical character. The nature of program algorithms has not changed at all in the past two decades, but IBM's position on the patent issues has completely reversed itself.)

The public interest representative on the Commission's working group on the computer program issues was William Keefauver, the lawyer who argued the *Benson* case before U.S. Supreme Court on behalf of AT&T (the assignee of Benson's patent rights). Keefauver has made no secret of the fact that he regards the Supreme Court's ruling that Benson's algorithm for converting binary coded decimals to pure binary form was unpatentable was wrongly decided. With Figueroa and Keefauver on the working group on the computer program-related issues, along with three other lawyers specializing in patent law (and an IBM attorney as an alternate member), it was widely expected that the group would conclude that patents were appropriate for computer program-related inventions. Indeed, any other conclusion would have been extremely surprising. (Samuelson has yet to meet a patent lawyer who has doubts about the advisability of patent protection for software innovations.)

Last spring the Commission published a set of questions for comment from the public. Most of the questions dealt with patent harmonization and other issues, but the first group of questions focused on the computer program-related issues. Even the manner in which the Commission stated its questions on the computer program issues suggested something other than an open mind on the issues. One of the questions, for example, asked whether there was any reason why patent protection should be removed for computer program-related inventions. This way of stating the question suggests the law clearly provided patent protection for computer program innovations when, in fact, the case law is in considerable disarray on this subject.

The Commission has acknowledged receiving 545 letters in response to this set of questions. Nearly 80% of the letters addressed the computer program-related questions; 60% addressed only the computer program-related issues. The Commission has not provided further information about the letters, such as the numbers of respondents who opposed or supported patent protection for computer program innovations. Electronic versions of some of these letters were posted on electronic bulletin boards. From these, it is clear that quite a number of the letters were critical of software patents and quite a number came from computing professionals.

The draft report of the Commission's working group on the computer program-related issues was released last January. Unsurprisingly, it concludes that patent protection for computer program-related inventions is well-established in the law and should be continued. By endorsing the view expressed some years ago by patent scholar Donald Chisum that algorithms and other computer program-related inventions are patentable because they are processes and have a technological character, the draft report seems to call (as Chisum also did) for the overruling of the 1972 *Gottschalk vs. Benson* decision in which the U.S. Supreme Court decision ruled that computer program algorithms were unpatentable on account of their mathematical character.

The draft report states that it considered all the letters submitted in the response to the request for public comments. But the report mainly mentions potential objections to the patenting of software innovations as a prelude to dismissing them. (This part of the report follows the form: "A" is not a problem because of X; "B" is not a problem because of Y; and so on.) The draft report does, however, recommend a number of changes in PTO procedures for dealing with program-related inventions. For example, it states that the Office should have better access to the prior art for software innovations and better ways of classifying software so that people can search more effectively for what has been patented before.

The draft report also asserts that Europe and Japan now strongly support patent protection for the patenting of computer program-related inventions, and that the major patent offices around the world are operating in substantial harmony concerning patent protection for software innovations. It further urges the U.S. to press those nations that do not provide patent protection for software innovations to modify their policies to make program-related inventions patentable, saying that the U.S. competitive edge in software depends on the availability of patent protection. (It would take an entire column to explain why the report's assertions about other nations' patent standards are not completely accurate, but it is worth noting that the competitive edge currently enjoyed by the U.S. software industry was achieved in a legal environment in which patent protection was not available for most computer program-related inventions.)

GATT-Related Developments

For the last several years, negotiations have been underway to reach agreement on international norms on intellectual property rights within the framework of the GATT. In mid-December 1991, a draft agreement on Trade Related Intellectual Property Rights (TRIPS) aimed at achieving this goal was distributed. It is now under consideration by member nations. Negotiations are expected to continue for some time. It is far from clear that this draft will be adopted, mainly because Third World and industrialized nations have not yet resolved some long-standing disagreements on a number of its provisions (such as those requiring patent or patent-like protection for new species of plants).

Only a few of the provisions of the draft TRIPS agreement deal with computer software issues. The main provision of the TRIPS agreement concerning intellectual property rights in computer programs is that which would require member nations to protect computer programs as "literary works" under copyright law. The patent section of the draft TRIPS agreement does not directly mention computer software, but the provision does say that patents are to be available without regard to the field of technology to which they pertain. Since it is difficult to dispute that computer programming pertains to a "field of technology," this provision can be interpreted as requiring member nations to protect software innovations by patent law (notwithstanding the statutory provisions that many nations have excluding many program-related inventions from patents and judicial interpretations in many nations that have tended to limit the extent of patent protection for software innovations).

Those who support this expansive interpretation of the draft TRIPS agreement, like those who wrote the Patent Advisory Commission draft report, tend to assert that there is already a significant consensus, at least among industrialized nations, in favor of patent protection for software innovations (when, in fact, there is not). They also tend to ignore significant differences in patentability standards employed by those nations that do provide some degree of patent protection for software innovations. At an international conference on software intellectual property rights sponsored by Japan's Software Information Technology Center held in Tokyo in December, Jean-Francois Verstrynge, the head of the EC Directorate which issued the EC Directive on Copyright Protection for Computer Programs, after listening to discussion of British, German, U.S. and Japanese patent caselaw on patent protection for computer program-related inventions, stated that the discussion had convinced him that it was premature to say there was sufficient consensus on this set of issues to make it part of the GATT framework.

Conclusion

The SIGGRAPH intellectual property rights survey, like the CHI '89 survey before it, demonstrates that there is strong support for copyright protection for source and object code, but strong opposition to extending copyright protection to such things as look and feel within these segments of the technical community. Those surveyed expected negative consequences for their own work and for the industry and community of which they were a part if the look and feel lawsuits established strong copyright protection for user interfaces. The survey also suggests there is a significant opposition within these communities concerning patent protection for software innovations.

Neither the SIGGRAPH or the CHI '89 surveys purport to be anything more than what they are: interesting sets of data about what people in these communities think about the legal issues that affect their field. It would be interesting to know whether these surveys accurately reflect the membership views of the ACM or the various SIGs. Survey respondents want ACM or their SIGs to get more involved in the legal issues. Some of this involvement might be educational in nature; some might be more proactive than that. Perhaps further surveys would be useful as well.

Intellectual property rights are, of course, not a popularity contest. What people in a particular field think the law should be on a particular issue, even if by substantial margins, does not necessarily mean the courts or the legislature will or should agree with that group's assessment. But what people think about the norms that will govern their work and the industry as a whole ought to matter (if for no other reason than if there is a substantial gap between what people in the field think the rule should be and what the rule is, they may not respect the rule, or may devise

strained interpretations of it that might lead to more litigation.) Resentment at being excluded from the process of shaping the rule can also undermine the effectiveness of a rule.

It may be, as some have suggested, that the disagreements that have arisen within the technical community about intellectual property rights issues are simply reflections of distress at the changing norms of the field. Just as the prairies of the western U.S. were once open to any traveler who might cross them or settle there, software was once developed in an environment in which intellectual property rights played a negligible or minor role. Just as the erection of fences marked the passing of the western frontier, the passage of the software copyright amendments and new interpretations of patent law that have expanded the role of patents in the protection of software innovations may simply be abstract legal equivalents to the barbed wire fences that brought about the end of the western frontier.

Computer programs are unquestionably an important item of commerce, not only in the U.S., but in many other nations. Given the international nature of commerce of this product and its associated services, it is understandable that the U.S. and other exporters of software products would press other nations for adoption of relatively uniform rules for protecting intellectual property rights in software. But it is a bad way for the U.S. (or any other country) to make public policy by pushing for adoption of an international treaty requiring member nations to give patent protection to software innovations and then use that requirement as a basis for asserting that the U.S. (or any other country) has to patent software innovations in order to comply with its treaty obligations.

Computing professionals rely on the strength of the software industry, both for their employment and for the tools with which they conduct their work. They have a strong and abiding interest in the success of this industry, and in the existence of intellectual property rights that provide needed incentives for investment in the industry. In addition, they have a strong sense of professional responsibility and they care very much about the norms that govern their work. By virtue of their experience in the field, computing professionals also have some insights about what kind and what extent of intellectual property protection for software is appropriate that those who are making policy in this area would do well to heed. ◘

Pamela Samuelson *is a Professor of Law at the University of Pittsburgh School of Law.* **Michel Denber** *is a researcher at Xerox Corporation's research facility in Rochester, New York.* **Robert J. Glushko** *is the President of Hypertext Engineering, Pittsburgh, Pennsylvania.*

It's completely rational.

Microsoft® C/C++ 7.0 now includes the Windows™ operating system version 3.1 SDK. For only $139, it's hard to believe this development kit is so complete. We're talking all the latest technology programmers have told us they want.

But you're still skeptical. You're asking yourself what is missing? Well, here's an admittedly long-winded list of what's included.

Object linking and embedding. Great for creating applications that share diverse types of data, like text and graphics. An application that supports OLE can cooperate with other OLE applications, even those from another vendor.

Pen computing APIs. Everything you need to create compelling applications for the next generation of pen computers. There are over 300 pages of documentation that give you the basic elements of Windows for Pen Computing.

Multimedia APIs. Our newest media control interface. You can incorporate animation, audio and video capability using third party devices and drivers. (It's the next best thing to virtual reality.)

Windows Debugging Version. Traps the most troublesome UAEs and helps you create more stable, robust applications. Contains GDI, KERNEL, and USER modules.

GUI Setup Toolkit. It will make hammering out a custom Windows-based setup program as simple as writing a script file.

Microsoft Foundation Classes. Now you can use the same building blocks we're using to build future versions of the Windows operating system. A rich set of 60 recyclable object classes provides logical order to more than 500 functions of the Windows API. Menus, GDI, OLE 1.0 and advanced diagnostics support are included. Tasks such as registering Windows' classes, building message loops, and managing device contexts are automatic.

Sample code. Over 75,000 lines.

Zoomin. A nifty tool for magnifying portions of a screen to help identify paint problems and other screen-related issues.

Heapwalker. Examine the global heap and local heaps used by active applications and dynamic-link libraries in your system.

Editors. And plenty of them. Dialog Editor lets you add, modify, and delete custom controls as you design and test dialog boxes on-screen. Image Editor modifies icons, bitmaps and cursors. Font Editor alters existing faces and creates new ones. Hotspot Editor creates and edits hypergraphics or bitmaps in one or more hotspots.

Spy Tools. Pssst. They make it possible to monitor Mouse, input/output from Windows, DDE and other messages between one or more Windows-based applications. You can also examine message parameter values.

Stress. Allows you to consume system resources for low-resource stress testing. Acquirable resources include the global heap, user heap, GDI heap, disk space and file handles. (Less job stress for you.)

Wdeb386. Now test and debug dynamic link libraries and Windows-based applications running in standard or 386-enhanced mode.

Multi-resolution bitmap compiler. Combine color and monochrome bitmaps with different resolutions into a single graphic.

Help Compiler. Create Help systems for applications that will take advantage of the new Windows 3.1 Help engine. Both context-sensitive and topical searches of Help files.

Compiler features. Unrestricted pre-compiled headers for speedy throughput. Explicit or automatic inlining of any C/C++ code for faster executions. Compressed code for size reductions.

CodeView® Debugger. This window-oriented debugger is a powerful, easy-to-use tool for analyzing MS-DOS® or Windows-based program behavior. Test the execution and examine data all at the same time. Display any combination of variables – global or local – while you halt or trace a program's execution. Versions for dual and single monitor debugging, even in a window, are supported.

Source Profilers. Optimize the performance of all your MS-DOS-based or Windows-based applications. Analyze your program to find code inefficiencies.

Whew! If you were to get all of this separately, it would cost many hundreds of dollars. And definitely will in the very near future.

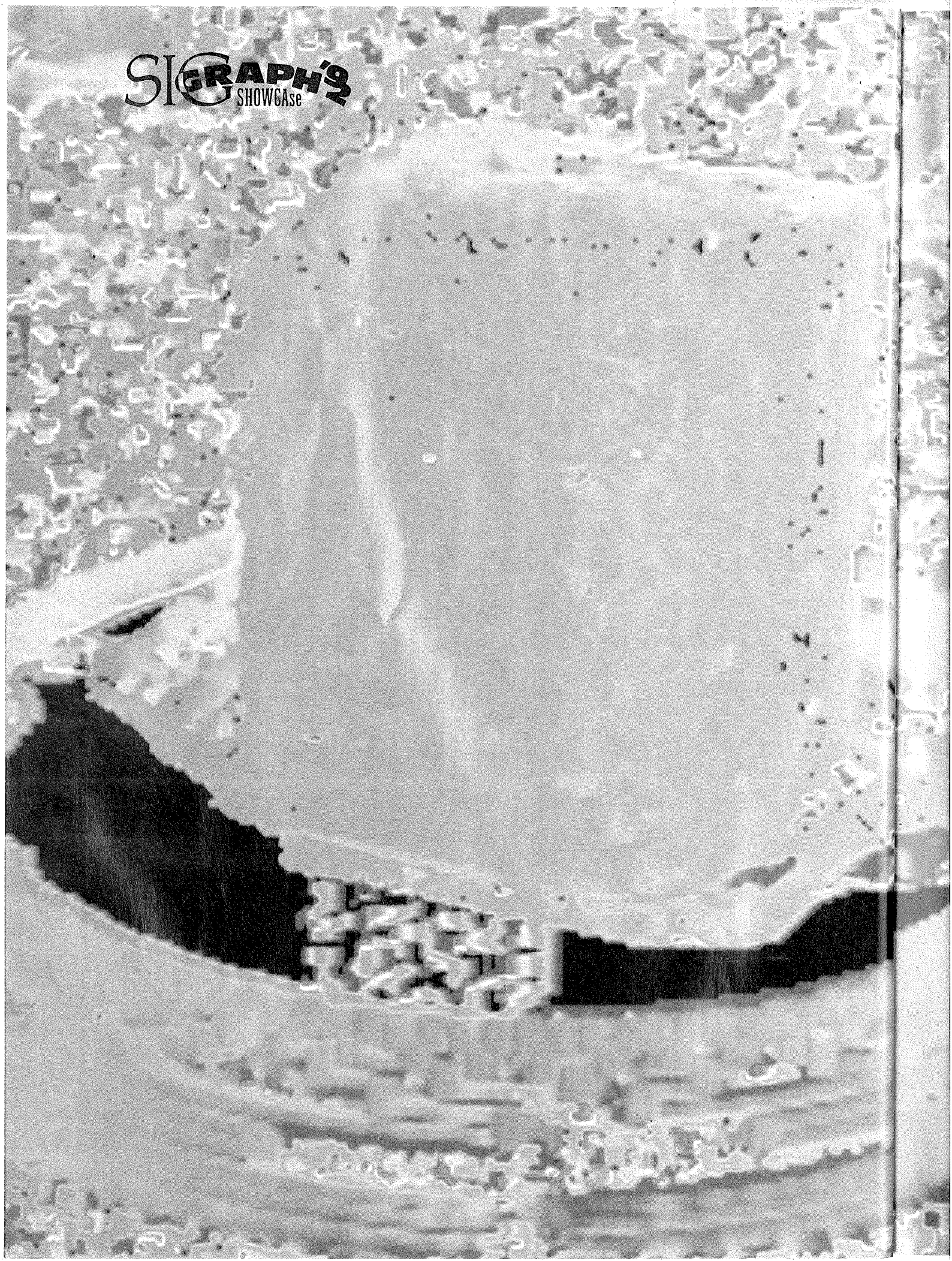
On the other hand, if you use a Microsoft or any other C/C++ product, and order an upgrade before June 30, 1992, it's yours for $139. That includes over 5,000 pages of C/C++ 7.0 documentation plus the entire Windows 3.1 SDK online documentation. We'll also throw in a free copy of Qualitas® 386-Max™ for MS-DOS-based development.

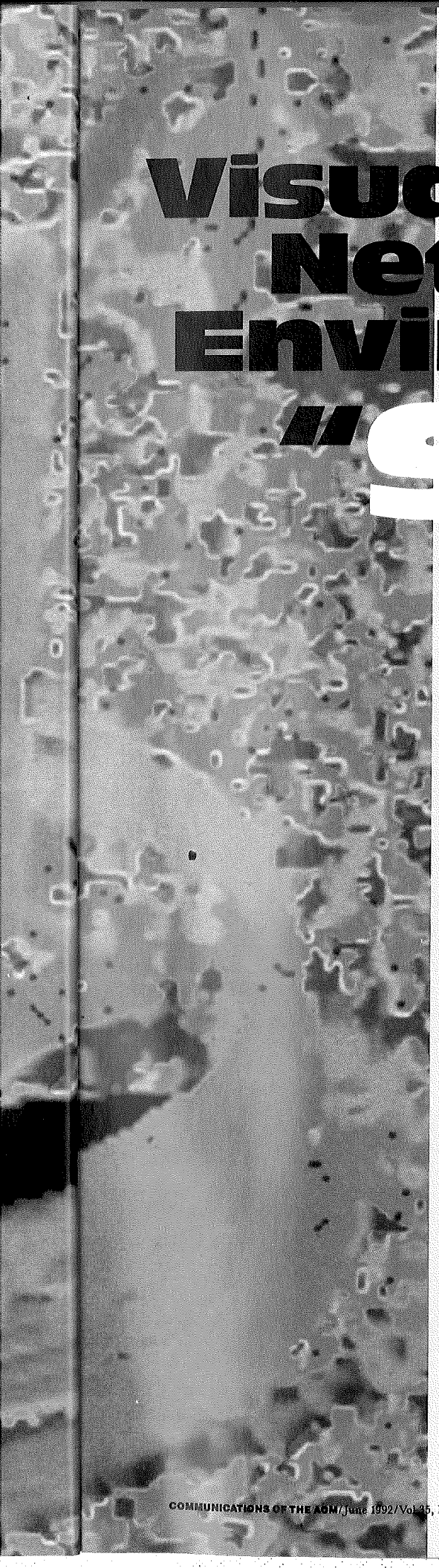
And if you want the Windows 3.1 SDK documentation in hard copy (over 5,000 pages), it's yours for an additional $150.

Look. You can defy the logic of thinking programmers everywhere and pass up this historic offer. Or you can call your reseller. Or just call us at (800) 541-1261, Dept. B21.

Microsoft®

It defies logic.


SIGGRAPH '92
SHOWCASE



Introduction

Visualization in Networked Environments

"Showcase" is a research exhibition and part of the ACM-sponsored SIGGRAPH '92 Conference to be held at Chicago's McCormick Place Convention Center from July 27 to 31. The exhibition will focus on the use of visualization in networked environments, particularly for interactively steering supercomputer applications and for facilitating collaboration between remote sites. "Showcase" projects are multiuser and interactive, providing both users and observers with the current state of the art in networked visual *computational science*. This term distinguishes the use of computers in the discipline sciences—physics, economics, medicine, astronomy, chemistry and biology—from computer science.

The network accessibility of Showcase follows in the footsteps of previous Supercomputing conferences. The Ultranet booth at Supercomputing '90 provided networked access to the Cornell National Supercomputing Facility and the entire exhibit floor of Supercomputing '91 was interconnected via a HIPPI network which in turn provided external access to the NSF-NET. Showcase will provide high-speed internal FDDI and external T3 network access to each of its projects for remote site communication. After SIGGRAPH, a portion of Showcase will travel to Minneapolis in November for Supercomputing '92.

Four distinct Showcase themes—local heterogeneous networked environments, remote visualization and collaboration paradigms, novel visual interfaces, and specific applications—are covered in this special section of *Communications*. In Metacomputing, Smarr and Catlett describe local networking environments made so transparent that users can create multicomputer applications as easily as single computer programs. Mercurio et al. wrap remote visualization, collaborative work and telepresence into the concept of "The Distributed Laboratory." A virtual reality interface is described in "The CAVE Audio-Visual Experience Automatic Virtual Environment." Finally, three novel applications of networked visualization (including a three-dimensional fax machine) are summarized in "Modeling and Analysis of Empirical Data in Collaborative Environments." For those who will not be able to attend SIGGRAPH '92, these four papers are a good representation of the kinds of research exhibits that comprise Showcase. For those who can attend, see you there.

Guest Editor

John C. Hart, Electronic Visualization Laboratory and National Center for Supercomputing Applications.

Metacomputing

Larry Smarr
Charles E. Catlett

From the standpoint of the average user, today's computer networks are extremely primitive compared to other networks. While the national power, transportation, and telecommunications networks have evolved to their present state of sophistication and ease of use, computer networks are at an early stage in their evolutionary process. Eventually, users will be unaware they are using any computer but the one on their desk, because it will have the capability to reach out across the national network and obtain whatever computational resources are necessary.

The computing resources transparently available to the user via this networked environment have been called a *metacomputer*. The metacomputer is a network of heterogeneous, computational resources linked by software in such a way that they can be used as easily as a personal computer. In fact, the PC can be thought of as a minimetacomputer, with a general-purpose microprocessor, perhaps floating point-intensive coprocessor, a computer to manage the I/O—or memory—hierarchy, and a specialized audio or graphics chip. Like the metacomputer, the minimetacomputer is a heterogeneous environment of computing engines connected by communications links. Driving the software development and system integration of the NCSA metacomputer are a set of "probe" metaapplications.

The first stage in constructing a metacomputer is to create and harness the software to make the user's job of utilizing different computational elements easier. For any one project, a typical user might use a desktop workstation, a remote supercomputer, a mainframe supporting the mass storage archive, and a specialized graphics computer. Some users have worked in this environment for the past decade, using *adhoc*, custom solutions, providing specific capabilities at best, in most cases moving data and porting applications by hand from machine to machine. The goal of building a metacomputer is elimination of the drudgery involved in carrying out a project on such a diverse collection of computer systems. This first stage is largely a software and hardware integration effort. It involves interconnecting all of the resources with high-performance networks, implementing a distributed file system, coordinating user access across the

various computational elements, and making the environment seamless using existing technology. This stage is well underway at a number of federal agency supercomputer centers.

The next stage in metacomputer development moves beyond the software integration of a heterogeneous network of computers. The second phase involves spreading a single application across several computers, allowing a center's heterogeneous collection of computers to work in concert on a single problem. This enables users to attempt types of computing that are virtually impossible without the metacomputer. Software that allows this to be done in a general way (as opposed to one-time, *ad hoc* solutions) is just now emerging and is in the process of being evaluated and improved as users begin to work with it.

The evolution of metacomputing capabilities is constrained not only by software but also by the network infrastructure. At any one point in time, the capabilities available on the local area metacomputer are roughly 12 months ahead of those available on a wide-area basis. In general, this is a result of the difference between the network capacity of a local area network (LAN) and that of a wide-area network (WAN). While the individual capabilities change over time, this flow of capabilities from LAN to WAN remains constant.

The third stage in metacomputer evolution will be a transparent national network that will dramatically increase the computational and information resources available to an application. This stage involves more than having the local metacomputer use remote resources (i.e., changing the distances between the components). Stage three involves putting into place both adequate WAN infrastructure and developing standards at the administrative, file system, security, accounting, and other levels to allow multiple LAN metacomputers to cooperate. While this third epoch represents the five-year horizon, an early step toward this goal is the collaboration between the four National Science Foundation (NSF) supercomputer centers to create a "national virtual machine room." Ultimately, this will grow to a truly national effort by encompassing any of the attached National Research and Education Network (NREN) systems. System software must evolve to transparently handle the identification of these resources and the distribution of work.

In this article, we will look at the three stages of metacomputing, beginning with the local area metacomputer at the National Center for Supercomputing Applications (NCSA) as an example of the first stage. The capabilities to be demonstrated in the SIGGRAPH '92 Showcase environment represent the beginnings of the second stage in metacomputing. This involves advanced user interfaces that allow for participatory computing as well as examples of capabilities that would not be possible without the underlying stage-one metacomputer. The third phase, a national metacomputer, is on the horizon as these new capabilities are expanded from the local metacomputer out onto Gbit/sec network testbeds.

LAN Metacomputer at NCSA

Following the PC analogy, the hardware of the LAN metacomputer at NCSA consists of subcomponents to handle processing, data storage and management, and user interface, with high-performance networks to allow communication between subcomponents (see Figure 1). Unlike the PC, the subsystems now are not chips or dedicated controllers, but entire computer systems whose software has been optimized for its task and communication with the other components. The processing unit of the metacomputer is a collection of systems representing today's three major architecture types: massively parallel (Thinking Machines CM-2 and CM-5), vector multiprocessor (CRAY-2, CRAY Y-MP, and Convex systems), and superscalar (IBM RS/6000 systems and Silicon Graphics (SGI) VGX multiprocessors). More generally, these are differentiated as shared memory. (Crays, Convex, and SGI) and distributed memory (CM-2, CM-5, and RS/6000s) systems.

Essential to the Phase I LAN metacomputer is the development of new software allowing the program applications planner to divide applications into a number of components that can be executed separately, often in parallel, on a collection of computers. This requires both a set of primitive utilities to allow low-level communications between parts of the code or *processes,* and the construction of a programming environment that takes available metacomputer resources into account during the design, coding, and execution phases of an application's development. One of the problems faced by the low-level communications software is that of converting data from one system's representation to that of a second system. NCSA has approached this problem through the creation of the Data Transfer Mechanism (DTM), which provides message-based interprocess communication and automatic data conversion to applications programmers and to designers of higher-level software development tools.[1]

At the level, above interprocess communication, there is a need for standard packages that help the applications designer parallelize code, decompose code into functional units, and spread that distributed application onto the metacomputer. NCSA's approach to designing a distributed applications environment has been to acquire and evaluate several leading packages for this purpose, including Parallel Virtual Machine (PVM),[2] and *Express,*[3] both of which allow the programmer to identify subprocesses or subsections of a dataset within the application and manage their distribution across a number of processors, either on the same

physical system or across a number of networked computational nodes. Other software systems NCSA is investigating include Distributed Network Queueing System (DNOS)[4] and Network Linda.[5] The goal of these efforts is to prototype distributed applications environments, which users can either use on their own LAN systems or use to attach NCSA computational resources when appropriate. Demonstrations in Showcase will include systems developed in these environments.

A balanced system is essential to the success of the metacomputer. The network must provide connectivity at application-required bandwiths between computational nodes, information and data storage locations, and user interface resources, in a manner independent of geographical location.

The national metacomputer, being developed on Gbit network testbeds such as the BLANCA testbed illustrated in Figure 2, will change the nature of the scientific process itself by providing the capability to collaborate with geographically dispersed researchers on Grand Challenge problems. Through heterogeneous networking technology, interactive communication in real time—from one-on-one dialogue to multiuser conferences—will be possible from the desktop. When the Internet begins to support capacities at 150Mbit/sec and above, commensurate with local area and campus area 100Mbit/sec FDDI networks, then remote services and distributed services will operate at roughly the same level as local services. This will result in the ability to extend local-area metacomputers to the national scale.

Metacomputing at SIGGRAPH '92 Showcase

The following descriptions represent a cross-section of a variety of capabilities to be demonstrated by metaapplication developers from many different institutions. These six applications also cut across three fundamental areas of computational science. *Theoretical simulation* can be thought of as using the metacomputer to solve scientific equations numerically. *Instrument/Sensor control* can be thought of as using the metacomputer to translate raw data from scientific instruments and sensors into visual images, allowing the user to interact with the instrument or sensor in real time as well. Finally, *Data Navigation* can be thought of as using the metacomputer to explore large databases, translating numerical data into human sensory input.

Theoretical Simulation

Theoretical simulation is the use of high-performance computing to perform numerical experiments, using scientific equations to create an artificial numerical world in the metacomputer memory where experiments take place without the constraints of space or time. One of these applications takes advantage of emerging virtual reality technologies to explore molecular structure, while a second theoretical simulation application we described allows the user to explore the formation and dynamics of severe weather systems. An important capability these applications require of the metacomputer is to easily interconnect several computers to work on a single problem at the same time.

Molecular Virtual Reality: This project will demonstrate the interaction between a virtual reality system and a molecular dynamics program running on a Connection Machine. Molecular dynamics models, developed by Klaus Schulten and his colleagues at the University of Illinois at Urbana-Champaign's Beckman Institute Center for Concurrent Biological Computing, are capable of simulating the ultrafast motion of macromolecular assemblies such as proteins.[6] The new generation of parallel machines allows one to rapidly simulate the response of biological macromolecules to small structural perturbations, administered through the virtual reality system, even for molecules of several thousand atoms.

Schulten's group, in collaboration with NCSA staff, developed a graphics program which collects the output of a separate program running on a Connection Machine and renders it on a Silicon Graphics workstation. The imagery can be displayed on the Fake Space Labs *boom* display system, VPL's EyePhone head-mounted display, or the Silicon Graphics workstation screen. The program provides the ability to interact with the molecule using a VPL DataGlove. The DataGlove communicates alterations of the molecular structure to the Connection Machine, restarting the dynamics program with altered molecular configurations.

This metaapplication will provide the opportunity to use Virtual Reality (VR) technology to monitor and control a simulation run on a Connection Machine stationed on the show floor. In the past, remote process control has involved starting, stopping, and changing the parameters of a numerical simulation. The VR user interface, on the

[1]DTM was developed by Jeff Terstriep at NCSA as part of the BLANCA Testbed efforts. NCSA's research on the BLANCA testbed is supported by funding from DARPA and NSF through the Corporation for National Research Initiatives.

[2]PVM was developed by a team at Oak Ridge National Laboratory, University of Tennessee, and Emory University. Also see A. Beguelin, J. Dongarra, G. Geist, R. Manchek, and V. Sunderam. Solving Computational Grand Challenges Using a Network of Supercomputers. In *Proceedings of the Fifth SIAM Conference on Parallel Processing*, Danny Sorenson, Ed., SIAM, Philadelphia, 1991.

[3]Express was developed at CalTech and is now being distributed by ParaSoft. It is a suite of tools similar to PVM.

[4]"DNQS, A Distributed Network Queueing System," and "DQS, A Distributed Queueing System," are both 1991 papers by Thomas Green and Jeff Snyder from SCRI/FSU. DNQS was developed at Florida State University.

[5]Network Linda was developed at Yale University.

[6]This research is by Mike Krogh, Rick Kufrin, William Humphrey and Klaus Schulten Department of Physics, National Center for Supercomputing Applications at Beckman Institute.

Figure 1.

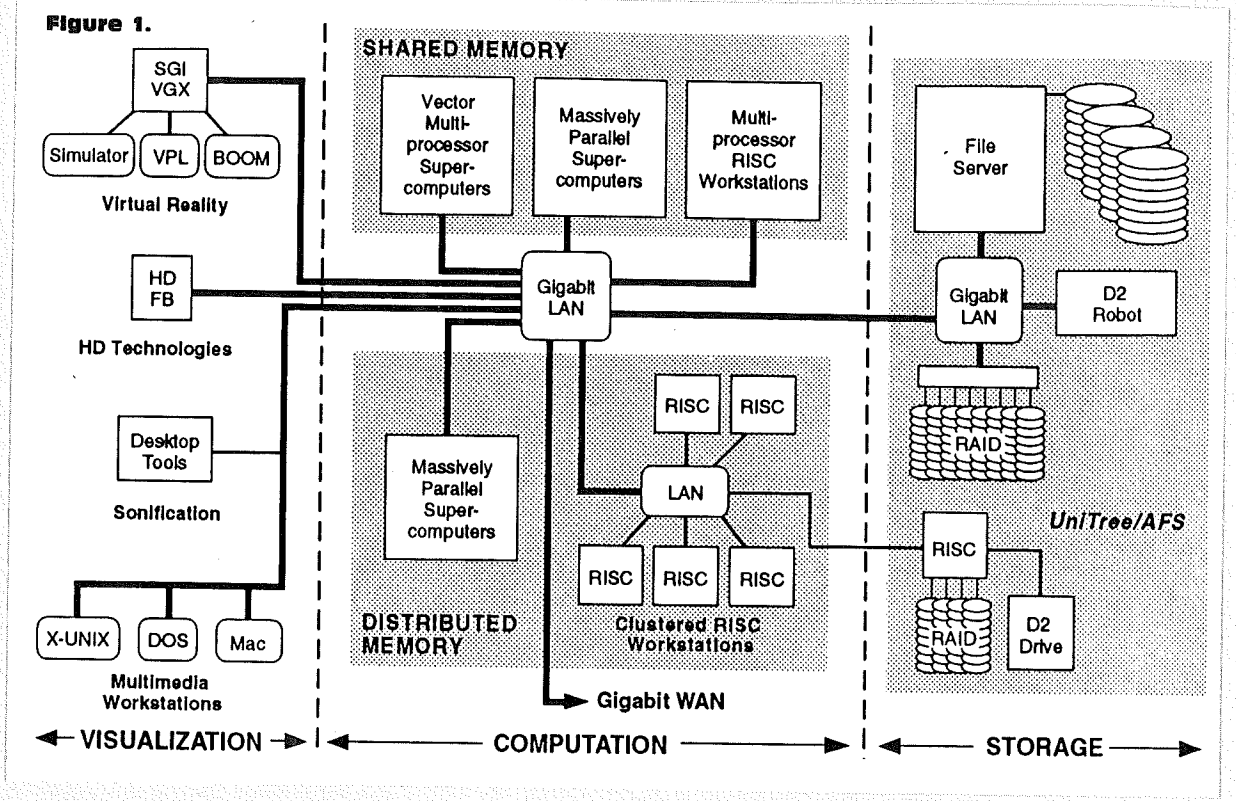


Figure 2.

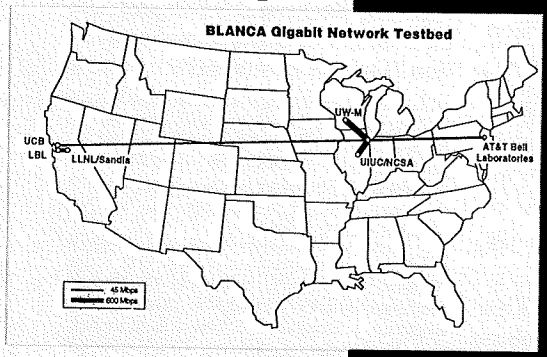


Figure 3.

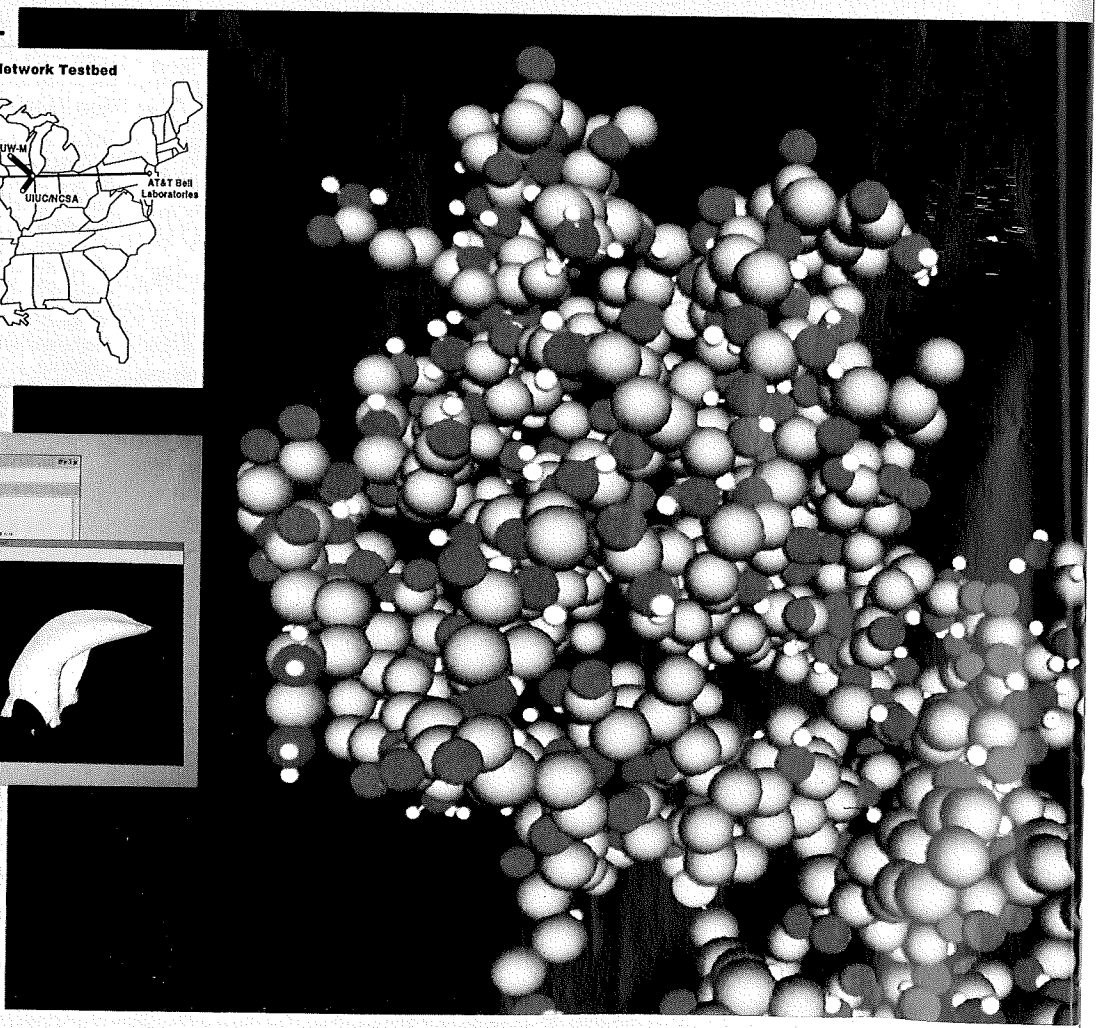
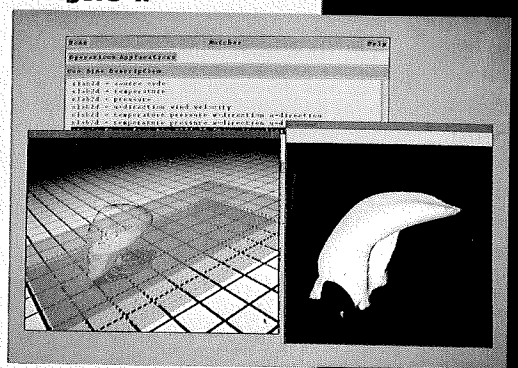


Figure 4.



other hand, allows the user to interact with and control the objects within the model—the molecules themselves—rather than just the computer running the model.

User-Executed Simulation/Analysis of Severe Thunderstorm Phenomena: In an effort to improve weather prediction, atmospheric science researchers are striving to better understand severe weather features. Coupled with special observing programs are intense numerical modeling studies that are being used to explore the relationship between these features and larger-scale weather conditions.[7] A supercomputer at NCSA will be used to run the model, and several workstations at both NCSA and Showcase will be used to perform distributed visualization processing and user control.

In Showcase, the visitor will be able to explore downburst evolution near the ground through coupled model initiation, simulation, analysis, and display modules. In this integrated, real-time environment, the analysis modules and visual display will be tied to new flow data as it becomes available from the model. This is a precursor to the kind of metacomputer forecasting environment that will couple observations, model simulations, and visualization together. The metacomputer is integral to the future forecasting environment for handling the large volumes of data from a variety of observational platforms and models being used to 'beat the real weather'. In the future, it is possible that real-time Doppler data will be used to initialize storm models to help predict the formation of tornadoes 20 to 30 minutes ahead of their actual occurrence.

Figure 1. LAN metacomputer at NCSA

Figure 2. BLANCA research participants include the University of California—Berkeley, Lawrence Livermore National Laboratories, University of Wisconsin-Madison (CS, Physics, Space Science and Engineering Center), and University of Illinois at Urbana-Champaign (CS, NCSA). Additional XUNET participants include Lawrence Livermore National Laboratories and Sandia. BLANCA uses facilities provided by the AT&T Bell Laboratories XUNET Communications Research Program in cooperation with Ameritech, Bell Atlantic, and Pacific Bell. Research on the BLANCA testbed is supported by the Corporation for National Research Initiatives with funding from industry, NSF, and DARPA. Diagram: Charles Catlett.

Figure 3. Three-dimensional image of molecule modeled with molecular dynamics software.
Credit: Klaus Schulten, NCSA visualization group

Figure 4. Comparing video images (background) with live three-dimensional output from thunderstorm model using the NCSA digital library.
Credit: Bob Wilhelmson, Jeff Terstriep

Instrument/sensor Control

Whereas the numerical simulation data came from a computational model, the data in the following applications comes from a scientific instrument. Now that most laboratory and medical instruments are being built with computers as control devices, remote observation and instrument control is possible using networks.

Interactive Imaging of Atomic Surfaces: The scanning tunneling microscope has revolutionized surface science by enabling the direct visualization of surface topography and electronic structure with atomic spatial resolution. This project will demonstrate interactive visualization and distributed control of remote imaging instrumentation.[8] Steering imaging experiments in real time is crucial, as it enables the scientist to optimally utilize the instrument for data collection by adjusting observation parameters during the experiment. A scanning tunneling microscope (STM) located in the Beckman Institute for Advanced Science and Technology at UIUC will be remotely controlled from a workstation at Showcase '92. The STM data will be sent as it is acquired to a Convex C3800 at NCSA for image-processing and visualization. This process will occur during data acquisition. STM instrument and visualization parameters will be under user control from a workstation at Showcase. The user will be able to remotely steer the STM in Urbana from Chicago and visualize surfaces at the atomic level in real time.

The project will use AVS, from AVS, Inc. to distributed components of the application between the Convex C3800 at NCSA and a showcase workstation. Viewit, a multidimensional visualization interface, will be used as the user interface for instrument control and imaging.

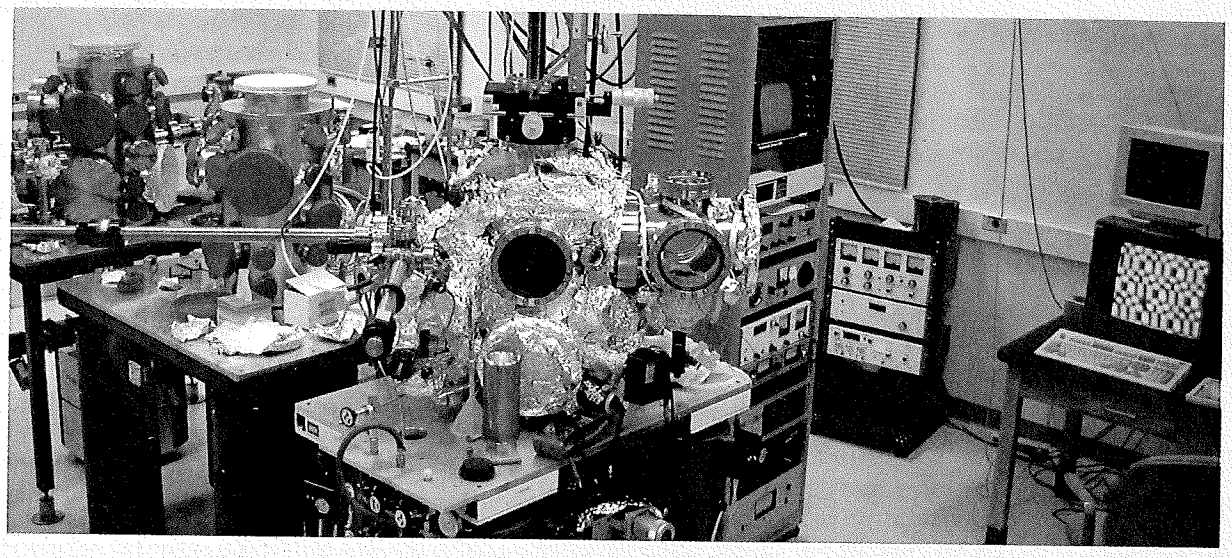
Data Navigation

Data navigation may be regarded not only as a field of computational science but as the method by which all computational science will soon be carried out. Both theoretical simulation and instrument/sensor control produce large sets of data that rapidly accumulate over time. Over the next several years, we will see an unprecedented growth in the amount of data that is stored as a result of theoretical simulation, instruments and sensors, and also textual and image data through network-based publication and collaboration. While the three previous applications involve user interfaces to specific types of data, the three following applications address the problem faced by scientists who are searching through many types of data. Capabilities are shown for solving the problem of locating data as well as examining the data.

Interactive Four-Dimensional Imaging: There are many different methods for visualizing biomedical image data sets. For instance, the Mayo Clinic Dynamic Spatial Reconstructor (DSR) is a CT scanner which can collect entire three-dimensional scans of a subject as quickly as 30 times a second. Viewing a study one two-dimensional

[7]This research is by Robert Wilhelmson, Crystal Shaw, Matthew Arrott, Gautum Mehrotra, and Jeff Thingvold, NCSA

[8]This research is by Clint Potter, Rachael Brady, Pat Moran, NCSA/Beckman Institute



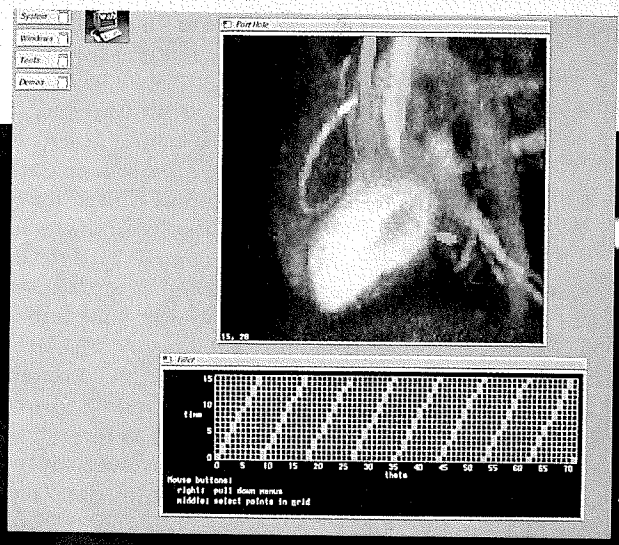


Figure 5.

Figure 6.

Figure 7.

plane at a time would take an enormous amount of time and still not directly reveal out-of-plane and temporal relationships.

The biomedical scientist requires computational tools for better navigating such an "ocean" of data. Two tools that are used extensively in the NCSA biomedical imaging activities are 'viewit' and 'tiller'.[9] 'Viewit' is a multidimensional "calculator" used for multidimensional image reconstruction and enhancement, and display preparation. It can be used to read instrument data, reconstruct, and perform volumetric projections saved in files as image frames. Each frame provides a view of the subject from a unique viewpoint at an instant in time. 'Tiller' collects frames generated by 'viewit," representing each frame as a cell on a two-dimensional grid. One axis of the grid represents a spatial trajectory and the other axis represents time. The user charts a course on this time-space map and then sets sail. A course specifies a frame sequence constructed on-the-fly and displayed interactively. This tool is particularly useful for exploring sets of precomputed volumetric images, allowing the user to move freely through the images by animating them.

At Showcase, interactive visualization of four-dimensional data will use an interface akin to that of 'Tiller'; however, the volumetric images will be generated on demand in real time, using the Connection Machine at NCSA. From a workstation at Showcase, the user will explore a large, four-dimensional data set stored at NCSA. A dog heart DSR data set from Eric Hoffman, University of Pennsylvania, will be used for the Showcase demo.

Figure 5. Scanning Tunneling Microscopy Laboratory at the Beckman Institute for Advanced Science and Technology.
Courtesy: Joe Lyding

Figure 6. Volume rendering sequence using "Tiller" to view dynamic spatial reconstructer data of a dog heart.
Credit: Pat Moran, NCSA

Figure 7. Three-dimensional rendering of Harvard CFA galaxy redshift data.
Credit: Margaret Geller, Harvard University, and NCSA visualization group

Scientific Multimedia Digital Library: The Scientific Digital Library[10] will be available for browsing and data analysis at Showcase. The library contains numerical simulation data, images, and other types of data as well as software. To initiate a session, the participant will use a Sun or SGI workstation, running the Digital Library user interface, to connect to a remote database located at NCSA. The user may then perform queries and receive responses from the database. The responses represent matches to specific queries about available data sets. After selecting a match, the user may elect to examine the data with a variety of scientific data analysis tools. The data is automatically retrieved from a remote system and presented to the researcher within the chosen tool.

One capability of the Digital Library was developed for radio astronomers. Data and processed images from radio telescopes are stored within the library and search mechanisms have been developed with search fields such as frequency and astronomical object names. This allows the radio astronomer to perform more specialized and comprehensive searches in the library based on the content of the data rather than simply by author or general subject.

The data may take the form of text, source code, data sets, images (static and animated), audio and even supercomputer simulations and visualizations. The digital library thus aims to handle the entire range of multimedia options. In addition, its distributed capabilities allow researchers to share their findings with one another, with the results displayed on multiple workstations—which could be located across the building or across the nation.

Navigating Simulated and Observed Cosmological Structures: The Cosmic Explorer[11] is motivated by Carl Sagan's imaginary spaceship in the PBS series "Cosmos," in which he explores the far corners of the universe. In this implementation, the user will explore the formation of the universe, the generation of astrophysical jets, and colliding galaxies by means of numerical simulations and VR technology. The numerical simulations produce very large data sets representing the cosmic structures and events. It is important for the scientist not only to be able to produce images from this data but to be able to animate events and view them from multiple perspectives.

Numerical simulations will be performed on supercomputers at NCSA and their resulting data sets will be stored at NCSA. Using the 45Mbit/sec NSFNET connection between Showcase and NCSA, data from these simulations will be visualized remotely using the VR 'CAVE'. The 'CAVE' will allow the viewer to "walk around" in the data, changing the view perspective as well as the proximity of the viewer to the objects in the data.

Two types of simulation data sets will be used. The first is produced by a galaxy cluster formation model and consists of galaxy position data representing the model's predicted large-scale structure of the universe. The second is produced by a cosmological event simulator that produces data representing structures caused by the interaction of gasses and objects in the universe.

[9]This research is by Clint Potter, Rachael Brady, Pat Moran, NCSA/Beckman Institute

[10]The digital library architecture and development work at NCSA is led by Jeff Terstriep.

[11]The Cosmic Explorer VR application software is based on software components already developed for VR and interactive graphic applications, including the Virtual Wind Tunnel developed by Steve Bryson of NASA Ames. Also integrated will be the software developed by Deyang Song and Mike Norman of NCSA for interactive visualization of numerical cosmology data bases, and the NCSA VR interface library developed by Mike McNeill.

Using the cosmic explorer and the 'CAVE', a user will be able to compare the simulated structure of the universe with the observed structure, using the Harvard CFA galaxy redshift database assembled by Margaret Geller and John Huchra. This will allow comparisons between the real and theoretical universes. The VR audience will be able to navigate the "Great Wall"—a supercluster of galaxies over 500 million light years in length, and zoom in on individual galaxies. Similarly, the simulated event structures such as gas jets and remains of colliding stars will be compared with similar structures observed by radio telescopes. The radio telescope data, as mentioned earlier, has been accumulated within the scientific multimedia digital library. This combined simulation/observation environment will also allow the participant to display time sequences of the simulation data, watching the structures evolve and converge with the observed data.

Acknowledgments

This article is in part an expanded version of the NCSA newsletter *Access* special issue on the metacomputer, Nov./Dec. 1991. Much information, text, and assistance was provided by Melanie Loots, Sara Latta, David Lawrence, Mike Krogh, Patti Carlson, Bob Wilhelmson, Clint Potter, Michael Norman, Jeff Terstriep, Klaus Schulten, Pat Moran, and Rachael Brady.

Further Reading

• Becker J. and Dagum L. Distributed 3-D particle simulation using cray Y-MP, CM-2. *NASA NAS, NASNEWS Numerical Aerodynamic Simulation Program Newsletter, 6,* 10. (Nov. 1992).

• Catlett, C.E. In search of gigabit applications. *IEEE Commun.* (Apr. 1992).

• Committee on Physical, Mathematical, and Engineering Sciences, Federal Coordinating Council for Science, Engineering, and Technology, Office of Science and Technology Policy. Grand challenges: High performance computing and communications. Supplement to the President's Fiscal Year 1992 Budget.

• Committee on Physical, Mathematical, and Engineering Sciences Federal Coordinating Council for Science, Engineering, and Technology, Office of Science and Technology Policy. Grand challenges 1993: High performance computing and communications. Supplement to the President's Fiscal Year 1993 Budget.

• Corcoran E. Calculating Reality. *Sci. Am. 264,* (Jan. 1991).

• Corporation for National Research Initiatives. 1991 Annual Testbed Reports. Reports prepared by project participants in each of five gigabit network testbeds.

• Hibbard W., Santek D., and Tripoli G. Interactive atmospheric data access via high speed networks. Computer Networks and ISDN Systems, 22, 1991, 103–109.

• Lederberg J. and Uncapher K. Towards a national collaboratory. Report of an individual workshop at the Rockefeller University, Mar. 1989.

• Lynch, D. Ed. *Internet System Handbook,* Manning Publications and Addison-Wesley, 1992.

• National Academy Press. Supercomputers: directions in technology and applications; [ISBN 0309-04088-4] Wash. D.C., 1989.

• *NCSA High-Performance Computing Newsletter.* NCSA's metacomputer: A special report. *access: 5,* 5 (Sept.–Dec. 1992)

• Smarr, L. Catlett, C.E., The Next Great Network *MIT Tech. Rev.,* July 1992.

• Stix G. Gigabit connection. *Sci. Am. 263* (Oct. 1990). C

CR Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications, Distributed databases;* C.3 [**Special-Purpose and Application-Based Systems**]; I.3.2 [**Computer Graphics**]: Graphics Systems—*Distributed network graphics, remote systems;* I.6.3 [**Simulation and modeling**]: Applications; J.2 [**Computer Applications**]: Physical Sciences and Engineering; J.3 [**Computer Applications**]: Life and Medical Sciences

General Terms:

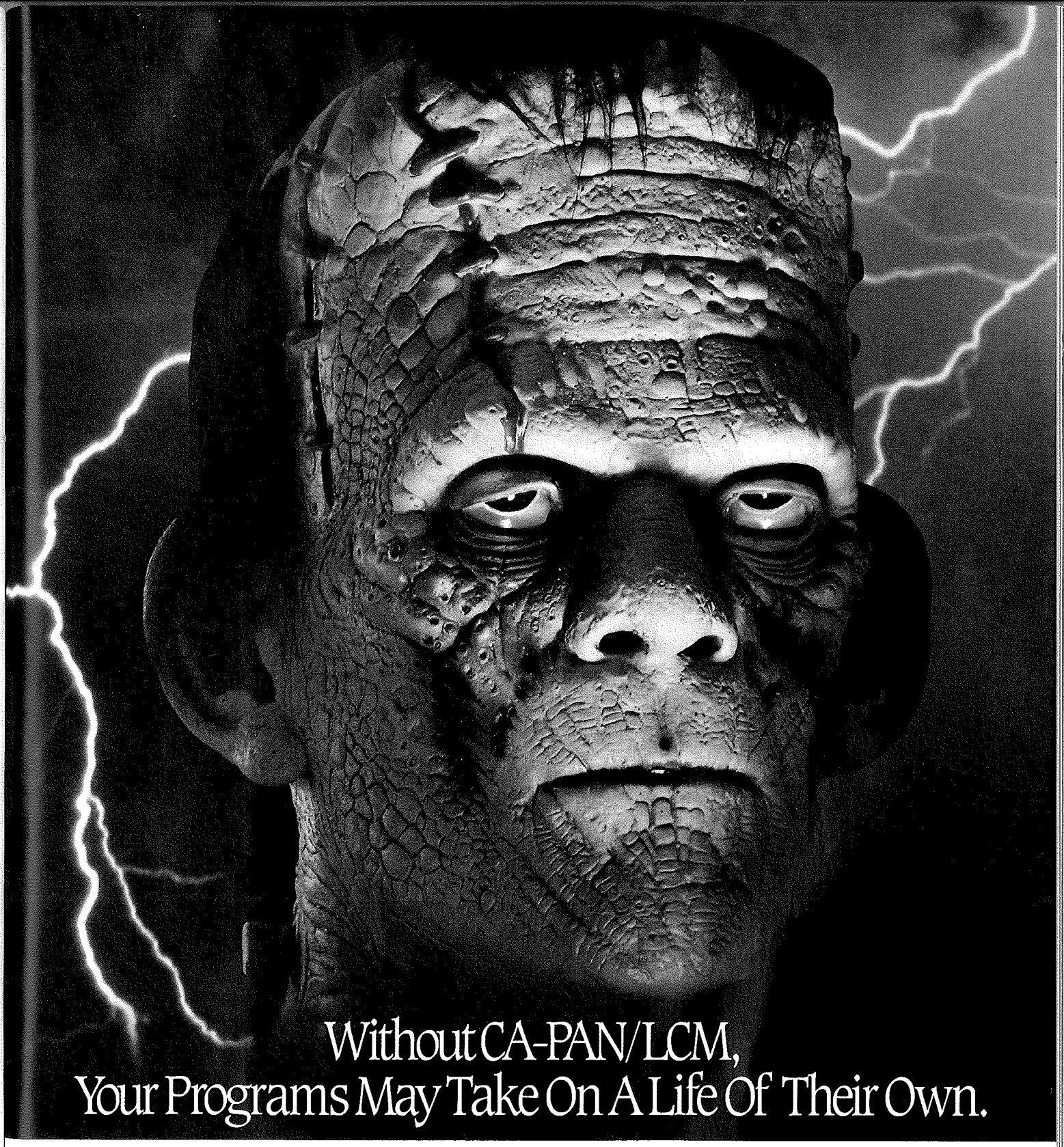
Additional Key Words and Phrases:

About the Authors:

LARRY SMARR is a professor of physics and astronomy at the University of Illinois at Urbana-Champaign (UIUC) and is the director of the National Center for Supercomputing Applications. His current research interests are in the investigation and visualization of astrophysical phenomena including black holes and the Higgs Field; email: smarr@ncsa.uiuc.edu

CHARLES E. CATLETT is associate director for computing and communications at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign. He is developing applications and programming environments for the BLANCA gigabit/second testbed. email: catlett@ncsa.uiuc.edu

Authors' Present Address: National Center for Supercomputing Applications, 605 East Springfield Ave., Champaign, IL 61820



Without CA-PAN/LCM, Your Programs May Take On A Life Of Their Own.

You've heard the horror stories.

Frightening tales of how uncontrolled change can wreak havoc in distributed application development environments.

CA's Computing Architecture For The 90s.

But CA-PAN/LCM™ puts those demons to rest.

It's the world's most advanced change and configuration management software, from Computer Associates.

"For large networked programming shops, CA-PAN/LCM was judged best...on the basis of multi-user security, ease of operation, and integration..."

PC Week, February 1991

Whatever your development environment—stand alone PC, LAN-based, or cooperative, workstation to mainframe—CA-PAN/LCM tells you the who, what, when, where and why of all changes made to any type of application component.

"CA-PAN/LCM represents a comprehensive solution to version control and configuration management, with special emphasis on security and data integrity."

Computer Language, August 1991

It also analyzes dependencies between application components and determines how to construct, maintain and enhance applications with a minimum number of steps.

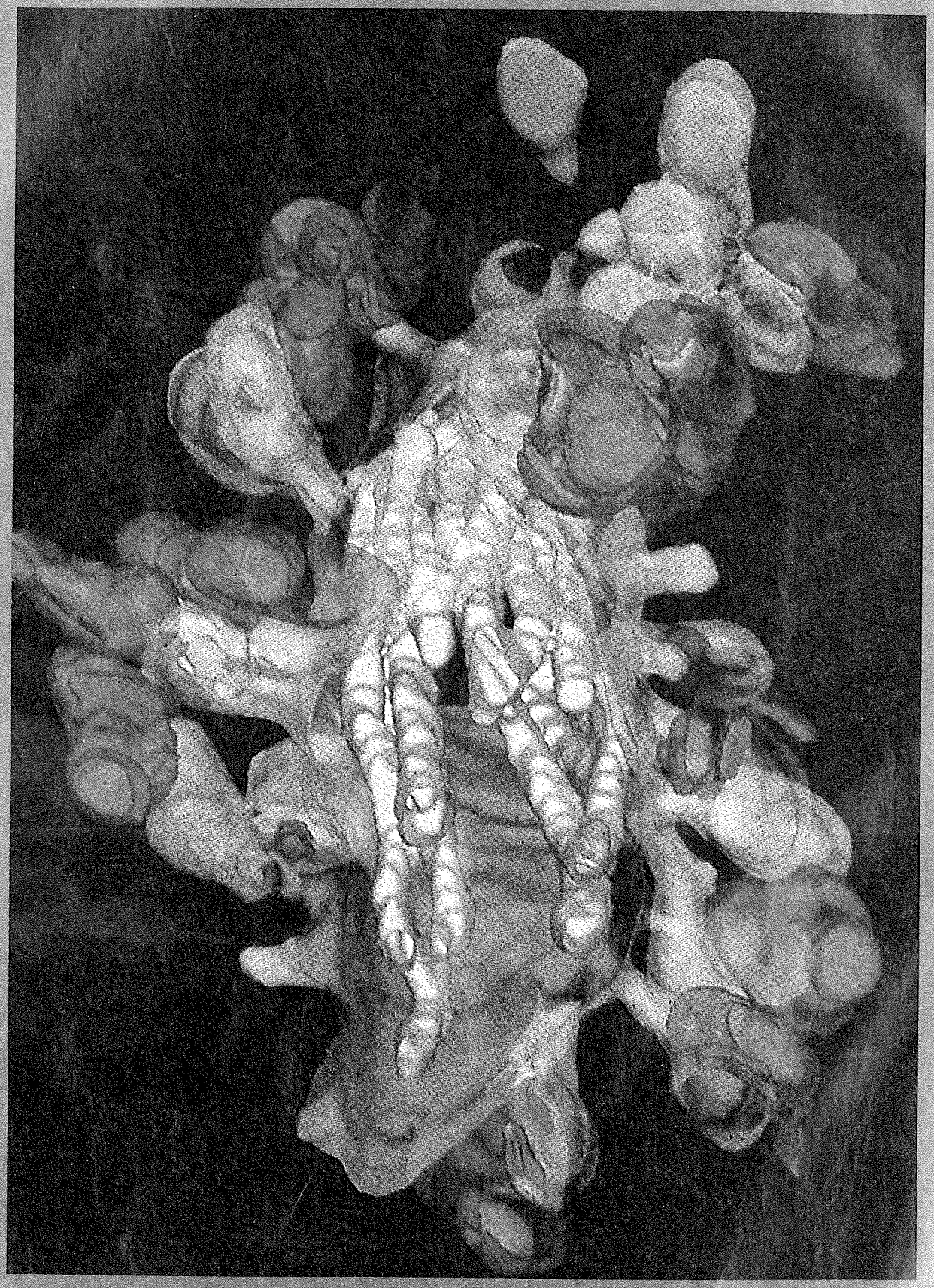
And CA-PAN/LCM even provides host integration to the most popular mainframe library control systems, ensuring cross-platform synchronization and data integrity.

Free Demo Disk

Dial 1-800-CALL CAI today, and we'll send you your free demo disk and a monster of a poster. But call right now.

After all, who knows what evil lurks in the hearts of your programs?

COMPUTER ASSOCIATES®
Software superior by design.



The Distributed Laboratory

An Interactive Visualization Environment for Electron Microscopy and 3D Imaging

Philip J. Mercurio
T. Todd Elvins
Stephen J. Young
Philip S. Cohen
Kevin R. Fall
Mark H. Ellisman

Visualization is often performed as a supplementary process to scientific investigation [2, 6, 10, 15, 17]. Presentation-quality renderings are produced from data that have been previously collected from instruments or computed via simulation. Even when interactive visualization environments are employed, they are typically used in a separate process that takes place after the experiments have been completed and the data have been manipulated into a form acceptable to the visualization tool. In this article we describe the *Microscopist's Workstation* (MWS) project. Its goal is to integrate the high-voltage electron microscope (HVEM) at the University of California at San Diego Microscopy and Imaging Resource (SDMIR) into a custom visualization application. Essentially, we are adopting the electron microscope to function as a computer peripheral.

The SDMIR microscope is a unique resource. By building an application around it, we hope to achieve two ends. Our primary result will be to extend the capabilities of the microscope. In studying researchers using the microscope, we have found that electron microscopes are used as cameras to collect data photographically for subsequent analysis. Although the SDMIR microscope is equipped for digital image acquisition, the images are still analyzed off-line, away from the microscope. By providing tools for image compositing, comparison, and morphometrics, and by enhancing, via stereopsis and computed tomography, the microscopist's perception of three-dimensional structures during the data collection process at the microscope, we hope to demonstrate a synergy between interactive visualization techniques and a sophisticated scientific instrument. The result should be a more powerful investigative tool.

Our secondary goal is to extend access to the microscope. The software for the MWS is designed to be

used either in the same room as the microscope or at a remote site, via a high-speed network. The software is also designed to be used either with the microscope on-line, or independent of the microscope, when the microscope is not available or not necessary for a particular task.

The United States Congress recently passed the High-Performance Computing and Communication Initiative, which will dedicate billions of dollars toward the development of supercomputing technology and the creation of a gigabit-per-second National Research and Education Network (NREN) by 1996. Such advances in network capabilities will facilitate higher-speed connections to unique facilities like SDMIR, and could lead to an extension of localized resources beyond geographical limitations. Providing network access to resources and custom software multiplies single resource, that are evolving into what we have been calling the *Distributed Laboratory.* While we are a long way from a completely remote-access SDMIR microscope, the Microscopist's Workstation prototype provides a glimpse of how high-speed networking can amalgamate distant sites into a highly interconnected research community.

The Distributed Laboratory

The MWS project is a collaborative effort among several research sites in San Diego: the San Diego Supercomputer Center, the Computer Science department and Medical School of the University of California, San Diego, and the Scripps Research Institute. Each institution is supplying hardware, networking, software development, and design expertise toward the project. One of the immediate spin-offs of this collaboration will be the establishment of higher-speed network connections between all of the sites.

The ongoing collaboration and discussions among the groups at different sites have lead us to begin to conceptualize a Distributed Laboratory that employs high-speed networks to integrate data acquisition, computational resources, visualization, and communication into a seamless, geographically distributed, investigative environment. As an example, consider the Internet. The Internet is not a single-governed entity but an amalgamation of networks, email routers, news distributors, and on-line databases that comprise an invaluable communication and information resource. The Distributed Laboratory would combine scientific facilities and connect researchers both within and across disciplines in the same way. The concept of a distributed laboratory leads to many areas of possible investigation, three of which are addressed by the MWS project: integrating a scientific instrument into the computing environment, seamless management of supercomputing resources, and multiuser collaboration.

Constructing an environment that accesses an instrument as a computer peripheral allows the device to be used by a researcher at a remote site with nearly the same efficiency available to an on-site researcher, that is, distributing the facility spatially. If the investigative software also operates with the instrument on-line and off-line, either independent of the instrument or with the instrument available for occasional "batch" processing, the facility is also distributed temporally. The result is the multiplication of single resources.

To present the researcher with a unified investigative environment, the management of computing resources must be automatic. Our experience with NetV software [7], which transparently distributes volume-rendering tasks to high-end resources, is a first step toward hiding the computing power behind the interface. In the MWS project, the workstation interface is the visible third of a triumvirate including the computer-controlled microscope and the supercomputing resources which are the remaining, hidden portions.

Focusing on bringing distributed data acquisition and computational resources to the workstation user interface allows the interface itself to be replicated among multiple researchers. This creates a collaborative research environment where two or more researchers can interact with one another and the instrument.

We consider the issues in developing a Distributed Laboratory approach to electron microscopy.

The SDMIR Electron Microscope

SDMIR is a new biology research facility established by the National Center for Research Resources of the National Institutes of Health, located in the Medical School at the University of California, San Diego. This laboratory was developed to support scientific research focused on relating biological function to structure, using state-of-the-art computer imaging and electron microscopy. The facilities of the laboratory are available to researchers throughout the U.S. The central instrument in the lab is a specially designed and equipped transmission electron microscope (TEM). A TEM resolves biological structures that are invisible when examined with a light microscope. It can magnify over 800,000 times and can resolve details in the 0.2 to 0.3 nanometer range. In contrast, a light microscope has a usable range of magnification up to about 1,000 times and a maximum resolution of approximately 200 nanometers. The resolution of a TEM is required, for example, to derive the "wiring diagram" of the nervous system. A light microscope provides a valuable panoramic view of the paths of nerve fibers from one area of the brain to another and of the jungle of nerve fibers arriving from different sources in the vicinity of the input sites of a particular nerve cell. But the resolution of a TEM is required to establish which fibers actually form contacts or synapses on the neuron as well as the particular form of the contact. In addition to studies concerning these

connections, scientists are currently using the electron microscope at SDMIR to examine the disruption of nerve cell components resulting from Alzheimer's disease [4, 13], the structural relations of protein molecules involved in the release of calcium inside neurons [12], and the three-dimensional form of the Golgi apparatus, where sugars are added to proteins. Examples of such three-dimensional reconstructions are shown in Figure 1.

As shown in Figure 2, many elements of the TEM are analogous to those of the light microscope. Instead of light, the specimen, which is placed in a vacuum column, is irradiated with a beam of electrons to form an image. Electrons emanating from a filament are accelerated toward the specimen. The electron beam is shaped by an electromagnetic condenser lens and directed by deflection coils to uniformly irradiate the specimen. Electrons penetrate the specimen and are magnified and focused by objective and projection electromagnetic lenses to form an image [1]. The image may be viewed directly on a phosphorous-coated screen, captured on film or digitized using a video system, and displayed on a video monitor. The contrast changes comprising the image formed by the microscope reflect variations in the electron densities within the specimen material. To visualize a particular component, the specimen must be processed to make the component more or less dense than the background. Preparing the specimen so that a particular biological component is differentiated, while its structural integrity is preserved, has been and continues to be the subject of much research.

The SDMIR instrument is one of the few high voltage transmission electron microscopes (HVEM) in the U.S. (see Figure 3). This instrument, a JEOL 4000EX, can employ a higher acceleration voltage (up to 400KV) than a conventional TEM (100KV). The higher accelerating voltage makes possible penetration and imaging of thicker specimens, up to 3 microns compared to 0.1 to 0.25 microns in the conventional TEM.

One focus of research at SDMIR is on the development of computer-assisted methods for deriving and exploring the three-dimensional structure of these thick sections [3, 10]. Because the HVEM has a very large depth of field, the image it forms is an orthogonal projection of the section. The depth dimension is lost. It is, however, possible to regain this dimension by obtaining images from different angles of view either by tilting the electron beam relative to the specimen or by mechanically tilting the specimen. In the simplest use of this principle, a stereo pair of images may be derived. In a more sophisticated approach, equivalent to that used in computer-assisted tomography (CAT), a three-dimensional volume of densities is derived from a series of images collected by successively tilting the specimen in small increments about an axis. This axial tomography procedure requires image processing to achieve proper alignment of the tilt series, and computationally intensive operations to derive the volume from the tilt series via filtered back-projection [9]. The SDMIR microscope has been specially designed to accomplish this task.

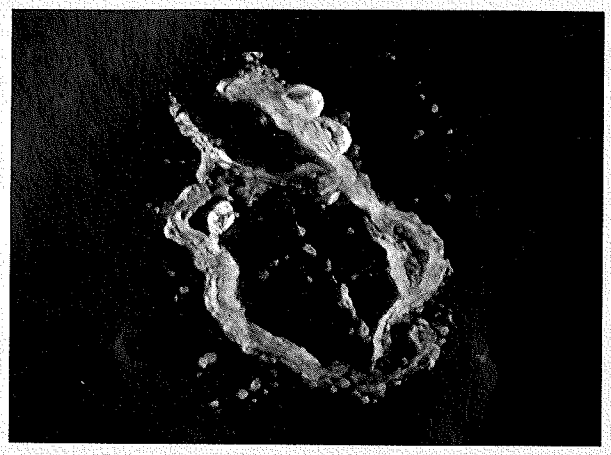
Special deflection coils make it possible to vary the beam entry into the material over sufficient angles to obtain pairs of images for stereo views. The objective lens was constructed with a larger gap into which the specimen is inserted, allowing rotation over the wider range of tilt angles required to perform axial tomography.

The HVEM is also well-equipped for automated control. It employs an A/D-D/A converter under microprocessor control to read, set, and display parameters such as acceleration voltage, magnification, focus, beam intensity, and astigmatic correction. These parameters can be manipulated with an array of knobs and buttons on the microscope. The parameters can also be set and read remotely with ASCII commands sent over an RS232 serial port. The specimen holder is mounted in a four-axis (X, Y, Z, and tilting) stage. The positions of the axes can be controlled manually and also by a dedicated IBM PC-AT compatible microcomputer driving four closed-loop motor-position encoders. The microcomputer program can be queried via a serial connection to set or retrieve the current stage position.

Microscope control and image acquisition are handled by a dedicated workstation called the Cap/Con (capture/control). The microscope and stage computer interact via RS232 lines with the Cap/Con, a Sun Microsystems SPARCstation 2 with several special purpose peripherals. Cap/Con also interfaces to an image processor that can acquire microscope images from both a real-time 512 × 512-8-bit video camera and a high-resolution 1,024 × 1,024-14-bit slow scan cooled CCD camera. Processed images may be viewed on a monitor at the microscope. In addition to providing views with better contrast than those seen by direct viewing at the microscope, the image processor may be used to display previously acquired images and stereo views. These capabilities make possible the design of the MWS.

The Microscopist's Workstation

We began our investigation into designing the MWS by talking to the scientists who constructed the SDMIR facility and used the microscope on a regular basis [15]. We also videotaped a microscopic session with an expert user, Tom Deerinck, who is adept not only with the SDMIR HVEM, but also with several other electron microscopes. We then transcribed and analyzed the videotape. The existing user interface to the microscope, consisting of the controls and displays on the microscope console, is the result of nearly five decades of development within the



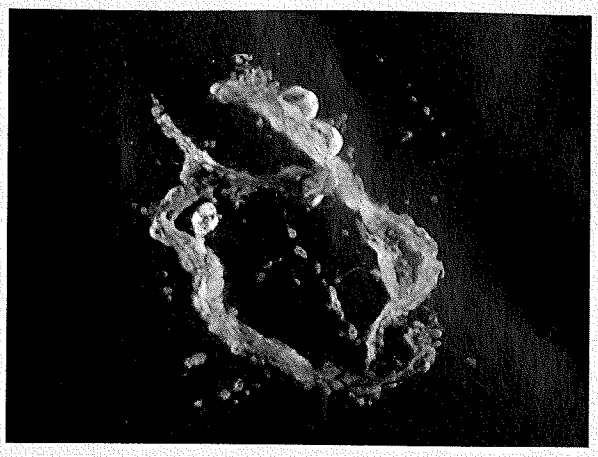
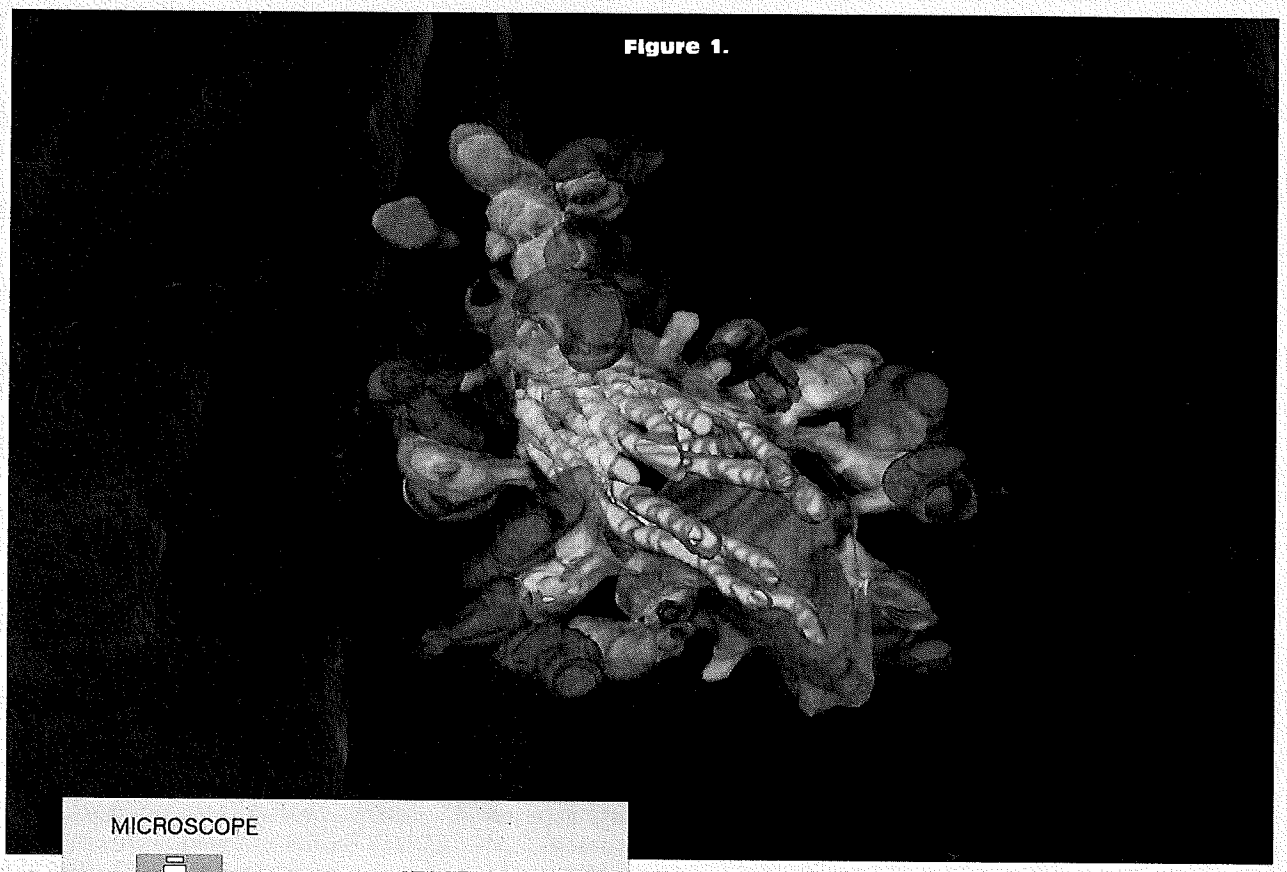


Figure 1.



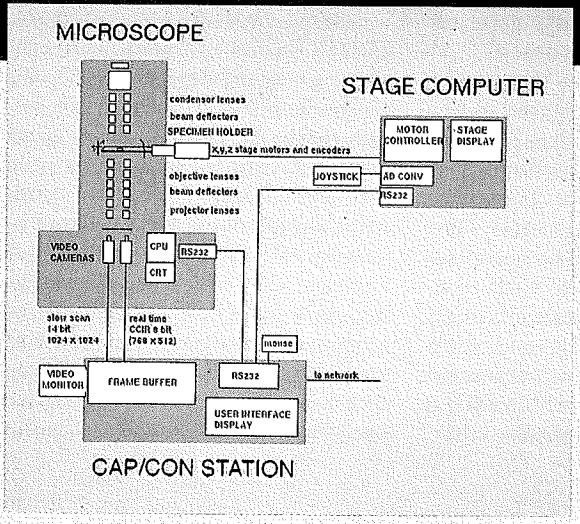


Figure 2.

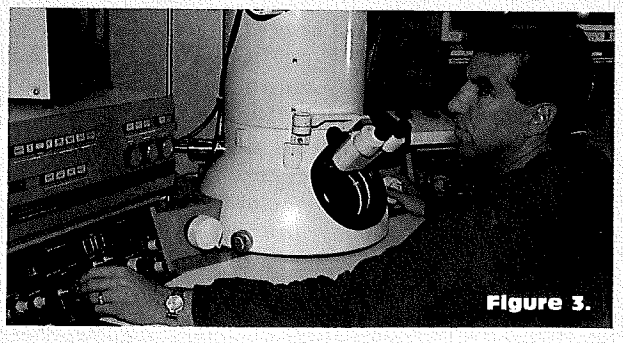


Figure 3.

electron microscopy industry, and already provides an excellent ergonomic environment for controlling the microscope. The user interviews and video analysis helped us design a visualization environment centered around users' needs when using the microscope as a research instrument, including tasks currently performed away from the microscope.

A microscope user begins by preparing a sample, usually by chemically staining the structures of interest, and depositing it on a grid, a 3mm diameter plate with a 2mm-diameter usable area. There are a wide assortment of grids available; some have a lattice of support wires and some are open throughout. One of the constraints is to be able to view a grid, in registration, from one microscopy session to the next. Our work, then, relies on grids that have registration marks at the center and at the edge. By calibrating to these marks, we can match the coordinate space of the software with that of the microscope across multiple sessions.

Much of the operation of the HVEM is already automated. The user has control over the X and Y stage position, the height and tilt of the stage, and parameters controlling the electron beam. The beam passes through the entire sample and is not focused on a particular plane, so the stage height is used only to control the axis of rotation for tilting the specimen. The primary positioning controls, then, are the X and Y stage controls. As previously mentioned, stage positions may be set manually or via the stage computer. The microscope computer display shows the coordinates in tenths of microns, in the range of ±1000 microns in X and Y.

There are three primary beam controls: magnification, which is measured in discrete steps from 100× to 800,000×, brightness, which is measured at the phosphor screen as current density, in units of picoamps per square centimeter, and the focus. In addition, astigmatic distortion can be corrected via stigmator coils and the beam can be shifted in X and Y as well as tilted up to five degrees. The beam controls operate in discrete steps so the user can return to a previous setting. This feature is important for an investigative tool and is incorporated into the design of the MWS software.

All the beam parameters must be adjusted to align and focus the image correctly, not only at the start of each session, but continually, as the microscope is operated. To operate the microscope remotely, a real-time video image of the microscope stage would need to be available to the remote microscopist, in addition to control over all of the beam parameters. Any perceivable lags or discontinuities in the video signal or in the transmission of the beam control information would make tuning the parameters extremely difficult. Since we cannot assume the availability of real-time video, the MWS software design is dependent on the presence of an operator at the microscope site, and includes support for interaction between the on-site operator and the remote researcher.

In studying users and the operation of the SDMIR laboratory, we found that electron microscopes are used mostly as cameras for collecting images either on film or in a digital format. For example, although the microscope's stage computer provides the ability to store several stage positions and return to them later, this capability may be used to record points of interest, but is rarely used to compare structures at different locations on the sample. Structure comparison can be better performed by taking photographs of each location and comparing them side-by-side, rather than by shuttling the stage back and forth. The MWS will extend the capabilities beyond those of a microscopic camera to allow image comparison and analysis while online with the microscope. Currently, tomography computation and display are performed away from the microscope. The workstation software will integrate the microscope's tomography capability into the existing microscopic environment.

Much of what is described in the Sidebar "A Look Forward" is incorporated into the design of the MWS prototype and will be demonstrated at the SIGGRAPH '92 Showcase. The major difference is that the operator on-site at SDMIR will have to be a highly trained electron microscopist, and will have to manually perform some of the tasks that will eventually become automated and incorporated into the Cap/Con software.

The scenario describes the remote researcher requesting a change in magnification by issuing a command that is sent to the Cap/Con workstation. At present, realigning and refocusing the electron beam after changing magnification requires a human's ability to navigate the multidimensional parameter space. The ideal focus is a subjective assessment; the researchers we have interviewed stated that the best images are often obtained by slightly under-focusing. There is no optimal focus condition applicable to every viewing situation, and no algorithmic means of finding an ideal configuration for all of the parameters for differing types of specimens. Automating focus adjustment is one of the challenges facing the electron microscopy field and is an active area of research. The MWS prototype will provide us with a platform on which to investigate approaches to solving this problem.

A human is also required in the computed tomography process, to center the structure of interest within the field of view and to ad-

Figure 1. Three-dimensional Golgi apparatus

Figure 2. A schematic of the microscope and interface

Figure 3. A high-voltage transmission electron microscope (HVEM)

A Look Forward

To make our design ideas for the Microscopist's Workstation more concrete, we began by postulating how a researcher in the near future might use the SDMIR facility remotely:

The year is 1998. Most major universities are connected via an Internet capable of delivering at least T3 speeds (45MB/sec) throughout the campus and across the country. Low-cost workstations on many researchers' desktops have the pixel-pushing capabilities, network connectivity, and compute power of the highest-quality workstations available in 1992. Demand for access to unique facilities like SDMIR has risen sharply—the microscope could be in use 24 hours a day, limited only by the logistics of getting the researchers in and out of the lab.

In an effort to maximize access to the HVEM, SDMIR has developed facilities for remote access, based on the prototype first shown at SIGGRAPH '92. Cap/Con, the dedicated image capture and microscope control workstation, connects to one or more workstations running the MWS software. There is one microscopist's workstation available for use at the SDMIR facility, but the distribution of tasks between Cap/Con and the local MWS is designed so that a remote user can access much of the functionality of the microscope by running the Microscopist's Workstation software on a distant workstation connected via a high-speed network. For compute-intensive calculations, such as the construction and rendering of volumes from tomography data, Cap/Con connects to a supercomputer at SDSC, which sends the rendered results either back to Cap/Con or on to the destination MWS (see Figure 4).

In preparation for using the microscope remotely, Perez, a hypothetical neuroscientist at a university in the midwest, has shipped her sample to UCSD. The night before her scheduled session, operators mounted her sample on a grid, loaded it, manually adjusted the focus and other parameters, and ran the *Survey* program on the Cap/Con. Survey uses the computer-controlled capabilities of the HVEM to scan the grid at a coarse magnification in a regular grid pattern, collecting a series of digital images. Because Perez's workstation is

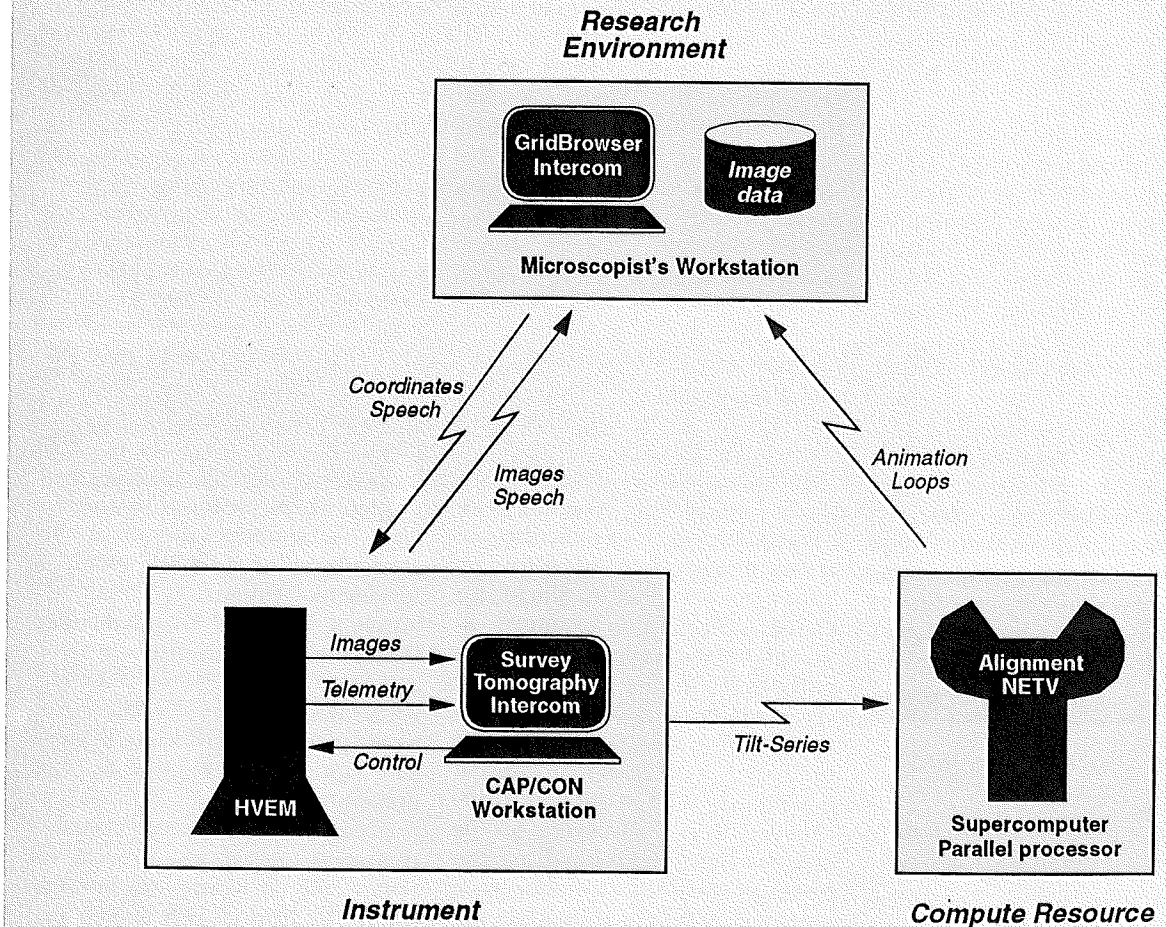


Figure 4. Simplified version of the system

equipped for stereopsis, the HVEM's beam-tilt capabilities are used to gather stereo images. Survey also performs the image-processing necessary to correct for edge distortions and nonuniform illumination, assembling the resulting images into two seamless montages, each 5K square pixels. For a 2mm grid, each resulting pixel will represent a 0.4-micron-square area of the sample. To facilitate interactive browsing of the data, reduced versions of the montages are also computed at several different resolutions, the smallest being a 512 × 512-pixel image. All of the bitmaps and the software Perez will need are sent via the Internet and arrive before she arrives at the lab in the morning.

Before starting her on-line session with the microscope, Perez first establishes a connection to SDMIR, where a skilled operator stands by to assist her. One of the SDMIR programs, *Intercom*, allows them to use the audio capabilities of their workstations to send segments of digitized speech back and forth over the network. Since they tend to spend long periods of time silently looking at material, it would be wasteful to keep a phone line open for the entire session. They may use the phone at various times during the session, however. Intercom collects a several-second segment of speech and sends it across the network. It does not attempt to handle continuous speech and thus is not subject to network latencies or interruptions.

Perez uses GridBrowser, another program provided in the SDMIR package to browse through the stereo montage data. GridBrowser has information about the coordinate space on which the grid was sampled; when Perez selects areas of interest it can record their coordinates in a form that can be sent back to the computer-control equipment on the microscope. GridBrowser allows her to pan across the entire bitmap using terrain management algorithms similar to those used in flight simulators to manage the portions of the bitmap currently in memory and maintain adequate update rates. GridBrowser also allows her to stimulate changing magnification by selecting among the precomputed reduced-scale images.

The SDMIR operator is also running a copy of the Microscopist's Workstation software. The two copies of software can interoperate via the network. At any given time, the two MWSs are either coordinated via a master/slave relationship, or running independently. When locked together, the slave MWS's display is controlled by input from the master MWS. Each displays a phantom cursor indicating the cursor position on the remote workstation. This extends the voice communication available via Intercom by adding deixis (pointing) [11].

Perez can request, via GridBrowser, that the microscope be set to a higher magnification. A reticle representing the size of the field of view of the microscope at the higher magnification appears as an overlay in GridBrowser. Perez can position the reticle to select a region of interest and can then request either a higher-magnification scan or a full three-dimensional rendering of some structure in view. If she requests a higher-magnification scan, GridBrowser sends the coordinates to Cap/Con, and also fills the display with the region of interest by pixel-replicating that portion of the local bitmap. The pixel-replicated image is replaced in a few seconds by the new data from the microscope. The Survey program running on Cap/Con collects and transmits a few fields from the area surrounding the selected region, in anticipation of the researcher's desire to roam around that area. If Perez chooses to zoom back out to the montage that was collected prior to the session, the higher-magnification dataset is retained on the MWS's local disk, and a marker will appear overlaid on the montage, indicating the availability of the additional data.

If Perez requests a three-dimensional rendering of the region of interest, GridBrowser transmits the request to Cap/Con and places a marker on the montage. Perez can continue to browse other areas of the montage, while Cap/Con begins the collection of the tomography images, a ±60 degree tilt in 2 degree increments. Cap/Con sends the resulting 60 slices to a supercomputer at SDSC for alignment and back-projection to produce a volumetric dataset. The volume dataset is rendered undergoing a rotation, for 72 frames, each fives degrees apart, and the resulting animation loop is sent to the remote MWS. When the animation arrives, GridBrowser changes the appearance of the market to indicate that Perez may now view the rendered images. The entire tomography-rendering-transmission process takes less than five minutes.

In addition to facilitating the browsing of images and managing requests to the microscope to gather further data, GridBrowser also provides image analysis tools. Perez can perform contrast enhancement and edge detection operations on images, along with other image processing operations, to aid in the detection of features. She can also perform morphometric analysis, measuring the extents and areas of features. At any time during her on-line session, Perez may request film or video hardcopy to be sent to her via surface mail [6].

Perez may also choose to perform her investigations without the microscope on-line. For example, if she has a sample that contained many similarly sized structures all of which could be identified in the 5K × 5K montage, she could use GridBrowser to mark each region of interest and submit the assembled coordinates to a queue at SDMIR for later processing. Since control of the microscope and image capture is fully automated, the main function of a human operator servicing the queue would be to load and unload each researcher's samples. A combination of on-line and off-line sessions is the most efficient use of the researcher's limited time, and keeps the microscope resource as busy as possible.

just the height of the stage, which controls the axis of rotation for tilting the stage. The remainder of the tomography process is currently semiautomated; we are attempting to automate more of the process.

Two MWS's based on SPARCstations are being assembled. An FDDI ring connects the SDMIR MWS and Cap/Con to SDSC, where they connect to the NSFNet T3 backbone. The MWS's each have a ViCOM VX graphics accelerator board, a tightly coupled ViCOM high-speed bulk memory board (128- to 256MB), and an FDDI interface. The VX accelerator is based on an Intel i860 processor along with 4MB of Intel RAM and a 16MB framebuffer. The bulk memory board connects to the VX via ViCOM's VXC Bus, at 320MB/sec. The resulting configuration affords us a large amount of image storage and manipulation capability.

Summary

Our goal in the MWS project is to investigate the potential advantages afforded by tightly coupling a scientific instrument under computer control with interactive visualization software. We are taking a user-centered approach to designing the software, in order to produce an environment that facilitates the researcher's interaction with electron microscopy data. This environment may also become available to users at remote sites, making the microscope a resource in a distributed laboratory.

Acknowledgments

The authors thank those whose support has contributed to this work, including the state of California, the University of California and the NSF Biological Sciences Directorate. Many thanks to Tom Deerinck and Steve Lamont at SDMIR, Jim Madden at UCSD, Tom Hutton, Paul Love and Mike Bailey at SDSC, and Tom Erickson at Apple Computer, Inc.

References

1. Agar, A., Alderson, R., Chescoe, D. Principles and practice of electron microscope operation. In *Practical Methods in Electron Microscopy,* A. M. Glauet, Ed., American Elsevier, N.Y., 1974.
2. Bancroft, G.V., Plessel, T., Merritt, F., and Walatka, P.P. Scientific visualization in computational aerodynamics at NASA Ames Research Center. *Computer 22,* 8 (Aug. 1989), 89–95.
3. Carragher, B., Hessler, D.F., Hinshaw, J.E., Martone, M., Milligan, R.A., Young, S.J., and Ellisman, M.H. Computer-aided image analysis and graphics in biological microscopy at higher voltages. In *Proceedings of the 49th Annual Meeting of the Electron Microscopy of America* (1991), p. 438.
4. Ellisman, M.H., Lindsey, J.D., Carragher, B.O., Kiyonaga, S.H., McEwen, L.R. and McEwen, B.F. Three-dimensional tomographic reconstructions of components of the Golgi apparatus imaged by selective staining and high voltage electron microscopy. *J. Cell Biology, III 5,* 2, 199a.
5. Ellisman, M.H., Carragher, B., and Martone, M. Three-dimensional microscopy and computer graphic representations of internal membrane system components of neurons. In *Proceedings of the 49th Annual Meeting of the Electron Microscopy of America* (1991), p. 150.
6. Elvins, T.T. A visualization computing environment for a widely dispersed scientific community. *State of the Art in Data Visualization,* ACM SIGGRAPH 90 Course Notes. Course Number 27 (Dallas, Tex., Aug. 1990), pp. 2.1–2.50.
7. Elvins, T.T., Nadeau, D. NetV: An experimental network-based volume visualization system. In *Proceedings of the IEEE Visualization '91 Conference,* IEEE Computer Society Press, Oct. 1991, pp. 239–245.
8. Frank, J., Radermacher, M. Three-dimensional reconstruction of nonperiodic macromolecular assemblies from electron micrographs. In *Advanced Techniques in Biological Electron Microscopy III,* J.K. Koehler, Ed., Springer Verlag Press, 1986.
9. Gore, A. HPCC policy champion forsees networked nation. *Commun. ACM 34,* 11 (Nov. 1991).
10. Haber, R.B., McNabb, D.A. Visualization idioms: A conceptual model for scientific visualization systems. In *Visualization in Scientific Computing,* Nielson and Shriver, Eds. IEEE Computer Society Press, 1990, pp. 74–93.
11. Hessler, D., Young, S., Carragher, B., Martone, M., Hinshaw, J., Milligan, R., Masliah, E., Whittacher, M., Lamont, S. and Ellisman, M. SYNU: Software for visualization of three-dimensional biological structures. *J. Microscopy Soc. Am. 22,* 1 (Mar. 1992).
12. Hill, W.C. and Hollan, J.D. Deixis and the future of visualization excellence. In *Proceedings of the IEEE Visualization '91 Conference,* IEEE Computer Society Press, Oct. 1991, pp. 314–319.
13. Martone, M.E., Zhang, Y., Simpliciano, V.M., Carragher, B.O., and Ellisman, M.H. Three-dimensional visualization of the internal membrane system of Avian Purkinje cell dendrites. *Soc. Neurosciences Abstracts* 1991, p. 1573.
14. Masliah, E., Ellisman, M., Carragher, B., Mallory, M., Young, S., Hansen, L., DeTeresa, R. and Terry, R. Three-dimensional analysis of the relationship between synaptic pathology and neuronal threads in Alzheimer Disease. *J. Neuropathology and Exp. Neurology,* (1992). To be published.
15. McCormick, B.H., DeFanti, T.S. and Brown, M.D. Visualization in scientific computing. *Comput. Graph. 21,* 6 (Nov. 1987).
16. Norman, D.A. and Draper, S.W., *User Centered System Design,* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1986.
17. Phillips, R.L. Distributed visualization at Los Alamos National Laboratory. *Computer 22,* 8 (Aug. 1989), 70–77.

CR Categories and Subject Descriptors: C.2.4 [**Computer-Communication networks**]: Distributed systems; I.3.2 [**Computer graphics**]: Graphics systems; I.3.6 [**Computer graphics**]: Methodology and techniques; J.3 [**Computer Applications**]: Life and medical sciences

General Terms: Design, Experimentation, Human Factors

Additional Key Words and Phrases: Distributed systems, microscopy, telepresence, visualization

About the Authors:

PHILIP J. MERCURIO is a staff programmer/analyst at the San Diego Supercomputer Center. His research interests include interactive scientific visualization, user interface design, stereoscopy, and virtual environments. **Author's Present Address:** San Diego Supercomputer Center, PO Box 85608, San Diego, CA 92186-9784. mercurio@sdsc.edu

T. TODD ELVINS is an associate staff programmer/analyst at the San Diego Supercomputer Center. His research interests include interactive scientific visualization, volume visualization, parallel computing, and network-based visualization tools. **Author's Present Address:** San Diego Supercomputer Center, PO Box 85608, San Diego, CA 92186-9784. todd@sdsc.edu

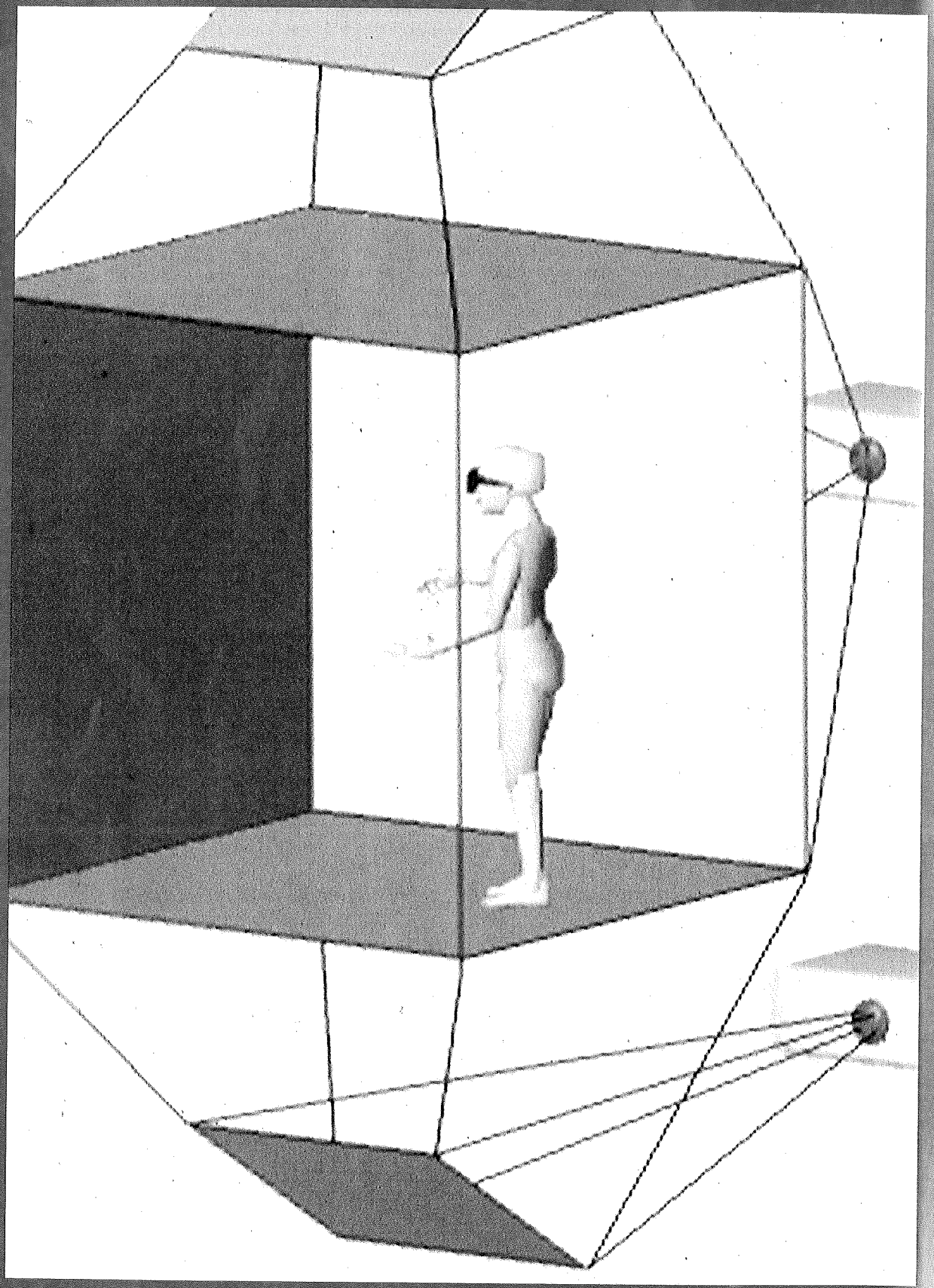
STEPHEN J. YOUNG is a specialist in the Department of Psychiatry and in the San Diego Microscopy and Imaging Resource in the Department of Neuroscience. His research interests are neurophysiology, neuropharmacology, and brain anatomy using computer-assisted methods for the analysis of brain function and neural structure. **Author's Present Address:** Department of Psychiatry, UCSD, La Jolla, CA 92093-0603 steve@alex.ucsd.edu.

PHILIP S. COHEN is is the director of Research Computing at The Scripps Research Institute (TSRI) in La Jolla, Calif. He is primarily interested in the application of computational methods and computer graphics in the biological sciences and chemistry. His department at TSRI is charged with keeping the advanced computational chemistry resources current. **Author's Present Address:** Research Computing Research Institute of Scripps Clinic, 10666 N. Torrey Pines Road, La Jolla, CA 92037. phil@scripps.edu

KEVIN R. FALL is a Ph.D. student in computer science and engineering at UCSD, and a networking consultant for the San Diego Supercomputer Center. His areas of research interest include networking and multimedia operating systems. **Author's Present Address:** San Diego Supercomputer Center, PO Box 85608, San Diego, CA 92186-9784. kfall@sdsc.edu

MARK H. ELLISMAN is a professor of neurosciences at the University of California at San Diego and is the director of the San Diego Microscopy and Imaging Resource. His research interests include investigation into the structure and function of the nervous system in health and disease as well as the development of new technologies for 3D Imaging with microscopes. **Author's Present Address:** San Diego Microscopy and Imaging Resource, UCSD, La Jolla, CA 92093-0608 mark@alex.ucsd.edu

This project is supported in part by the following: NSF grants ASC8902825 and ASC9008413; the San Diego Supercomputer Center; and DIR8822633, DCB8811713, DCB8819423; as well as NIH grants HL27470, NS14718, NS26739, RR04050 and M.H.E. Additionally, M.H.E. is the recipient of a Senator Jacob Javits Neuroscience Investigator Award from the NINDS, which is also supporting this work.



The Cave

Audio Visual Experience Automatic Virtual Environment

Carolina Cruz-Neira
Daniel J. Sandin
Thomas A. DeFanti
Robert V. Kenyon
John C. Hart

The CAVE is a new virtual reality interface. In its abstract design, it consists of a room whose walls, ceiling and floor surround a viewer with projected images. Its design overcomes many of the problems encountered by other virtual reality systems and can be constructed from currently available technology. Suspension of disbelief and viewer-centered perspective, are often used to describe such systems.

Suspension of Disbelief: This term arose from film criticism and is defined as the ability to give in to a simulation—to ignore its medium. The early attempts of the entertainment industry to achieve better suspension of disbelief laid the foundations for current virtual reality research. Suspension of disbelief is a fundamental part of the effective use of a virtual reality interface. Until we can ignore the interface and concentrate on the application, virtual reality will remain a novel experience instead of a serious visualization tool.

Viewer-Centered Perspective: The perspective simulation of common visualization systems dates back to the Renaissance, and is based in a mythical camera positioned along an axis extended perpendicular from the center of the screen. Viewer-centered perspective simulates the perspective view from the location of the viewer. To maintain correct perspective, a sensor that continuously reports the viewer's position to the simulation is commonly used. Without this perspective, the viewer becomes less a part of the environment, and a full suspension of disbelief becomes increasingly difficult.

Paradigms

Research in virtual reality began in 1965, when Ivan E. Sutherland proposed the "Ultimate Display," which would completely override the user's senses, totally immersing the user in the computer simulation [16]. Modern virtual reality research has split into four distinct directions, based primarily on differences in display devices.

Cathode Ray Tube (CRT): This is the simple monitor and is the most basic visual paradigm for virtual reality, though other kinds of monitors are also used. The most

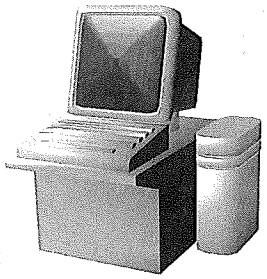
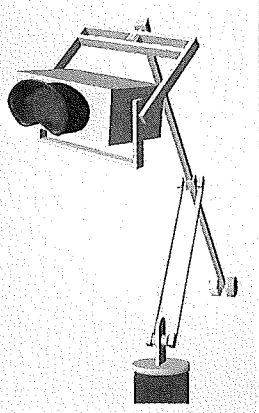
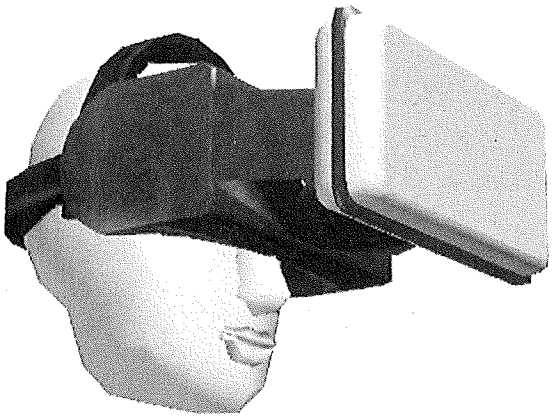


Figure 1.
Figure 3.





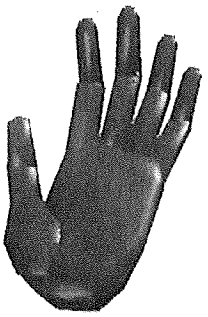
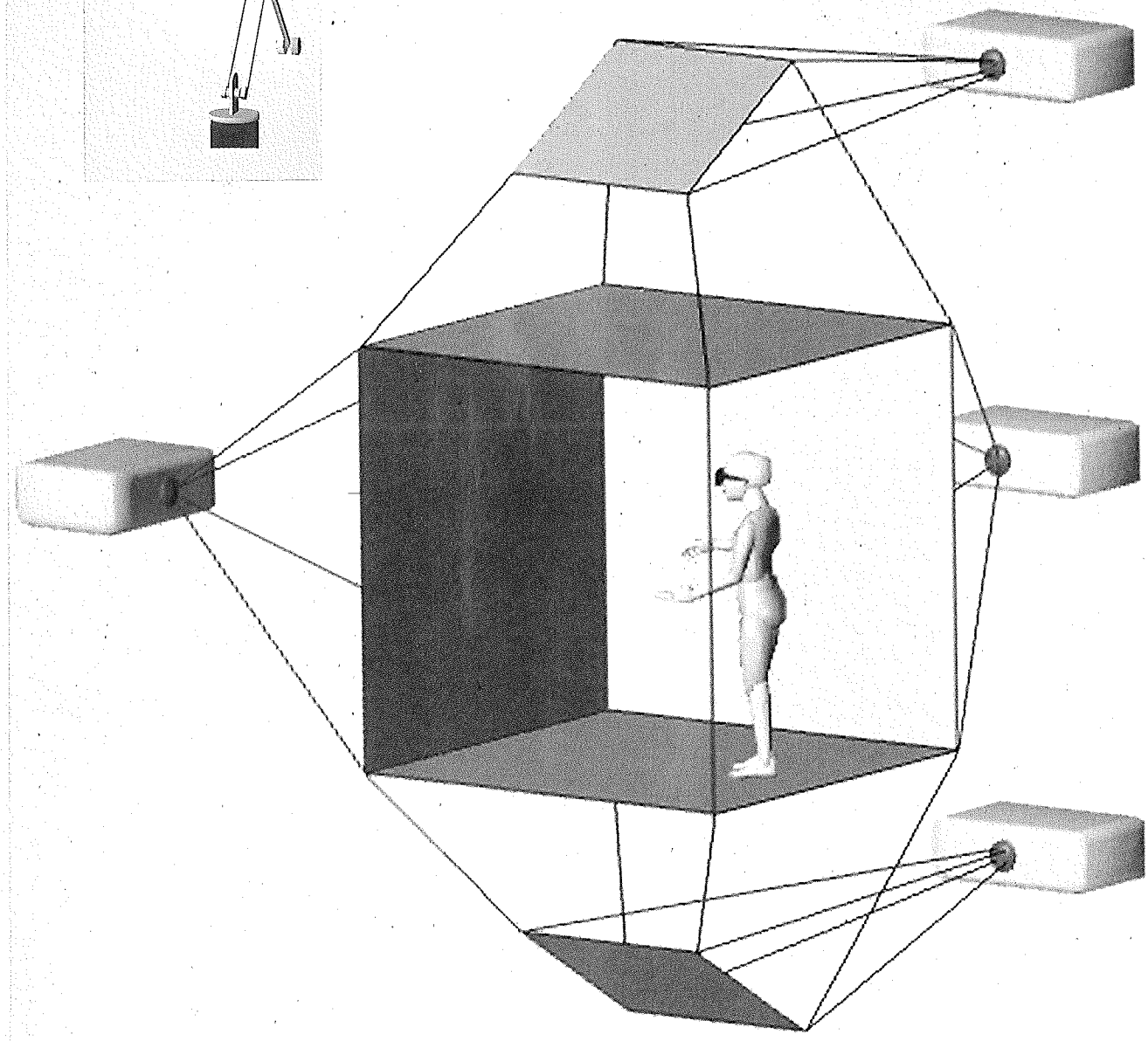


Figure 2.
Figure 4.



broadly used system is a small monoscopic display whose perspective is based on the Renaissance mythical camera model. This minimal visual interface may be enhanced for virtual reality use with the addition of stereo and viewer-centered perspective [10].

Head-Mounted Display (HMD): This is one of the most popular virtual reality visual interfaces. It consists of a stereo pair of small displays that cover the eyes. A head-tracking device provides the location and orientation of the viewer to simulate the correct view [4, 5, 11, 17].

Binocular Omni-Oriented Monitor (BOOM): Like the head-mounted display, the BOOM mounts small displays in front of the eyes, though more like binoculars than like goggles. Unlike the head-mounted display, the BOOM is suspended from an articulated arm, which measures its position and orientation in space and counterbalances its mass. Moreover, the user is expected to hold and position the BOOM manually throughout the virtual reality experience.

Audio-Visual Experience Automatic Virtual Environment (CAVE): This is the fourth visual paradigm for virtual reality and is a recursive acronym, also reminiscent of Plato's allegory of the cave [12]. The CAVE is a cube with display-screen faces surrounding a viewer. It is similar to surround systems such as OMNIMAX theaters and early flight simulators [14, 15]. Its more recent instance is coupled with a head-tracking device. As the viewer moves within the bounds of the CAVE, the correct perspective and stereo projections of the environment appear on the display screens.

Figure 1. Cathode ray tube (CRT)

Figure 2. Head-mounted display (HMD)

Figure 3. Binocular-omni-oriented monitor (BOOM)

Figure 4. CAVE Audio visual experience automatic virtual environment (CAVE)

Immersion Issues:

Immersion is the degree of visual simulation a virtual reality interface provides for the viewer—the degree of the suspension of disbelief. The five main issues in creating a powerful suspension of disbelief are shown in Table 1.

Field of View: The field of view represents the visual angle a viewer can see without head rotation. The simplest formulation, using W as the width of the display and D as the distance from the viewer to the display, is derived as a single angle

$$\theta = 2 \tan^{-1} \frac{W}{2D}$$

which describes the horizontal visual angle.

The field of view is variable in the CRT paradigm and is based on the size of the CRT and the viewer's distance from it. Viewing a 19-inch diagonal CRT from 18 inches produces a 45° field of view. The field of view for each eye is fixed in the BOOM and HMD paradigms. In the HMD paradigm, field of view angles from 100° to 140° are common. The Fake Space BOOM allows the viewer to adjust the field of view anywhere from 90° to 120°.

The field of view of the CAVE display varies by viewer location for each individual screen but achieves a full 360° for the entire display. Hence, the viewer experiences a maximal field of view, which may be limited by display hardware such as stereo glasses.

Cinerama and IMAX theaters fall in the CRT interface paradigm. These noninteractive virtual reality visual interfaces create a larger field of view by increasing the size of the projection screens. Cinerama used three vertical-edge linked projection screens and three synchronized projectors, whereas IMAX uses one very large screen which is placed near the viewing audience. The success of both of these systems in enhancing the suspension of disbelief characterizes the dependence of virtual reality on field of view.

Panorama: This is the ability of a display to surround the viewer and is crucial in creating a sense of immersion. It differs from field of view in that head rotation is used to view panorama. The CRT paradigm is not well-suited for panorama, and is generally treated as a window into some virtual environment. The OMNIMAX theater creates a sense of panorama by placing a large hemispherical screen about the viewer. The sense of panorama is strong in the BOOM and HMD interfaces, simulating everything the viewer sees. Viewer rotation is fast and smooth in the BOOM interface, due to its mechanical rotation-sensing equipment, though the inertia of the BOOM limits the rate of rotation.

The HMD is light, compact and easy enough to move quickly. Hence, the viewer can alter position and orientation much faster than present day tracking equipment. The result is a distracting lag: when the user turns, the environment turns with the user and then moves back to the correct orientation. Users of such systems are forced to move quite slowly and smoothly to avoid this problem.

The CAVE solves this problem by showing all views from a fixed location simultaneously. Users of the CAVE experience the same viewer location and head rotation measurement delays as do users of the HMD, but since rotations only require a small alteration to the stereo projections the effect is less noticeable.

Viewer-Centered Perspective: This depends heavily on the speed and accuracy of viewer location sensing. The viewer-centered perspective of each of the CRT, HMD and CAVE environments suffers from the same delay problems due to slow viewer location sensing. Hence, viewers compensate by moving slowly. The mechanical

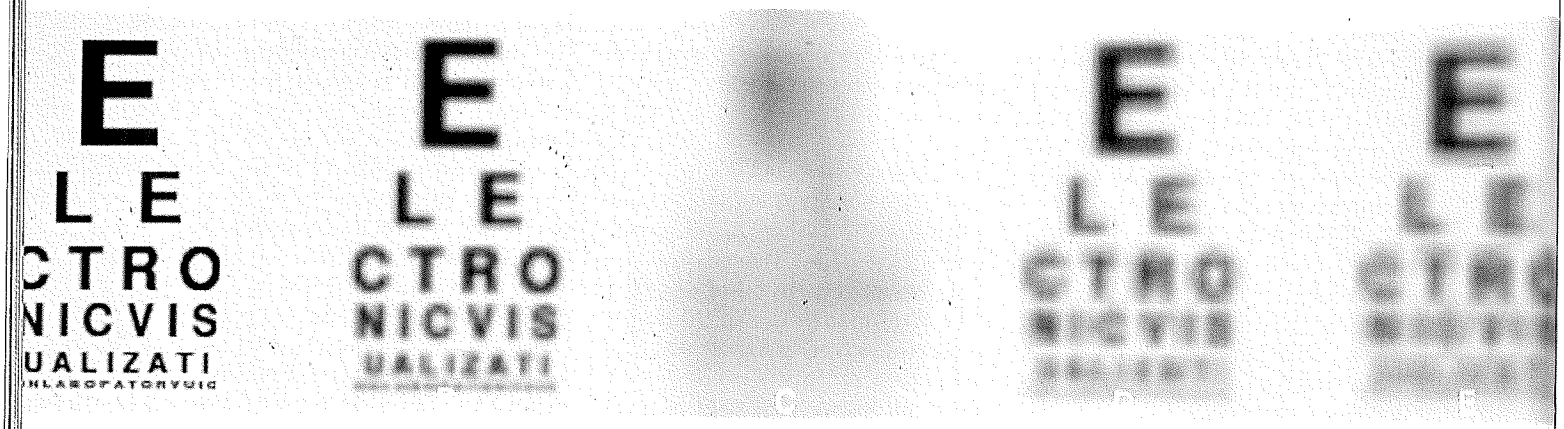


Figure 5. Representation of what viewer would see if he or she had visual acuity of a) 20/20, average eyesight b) 20/40, CRT interface c) 20/85, the Boom interface d) 20/425, the HMD interface e) 20/110, the CAVE interface. The lines on the eyechart are from top to bottom, 20/200, 20/100, 20/80, 20/60, 20/40, 20/20.

position-sensing of the BOOM overcomes this delay with faster location reporting at the expense of high inertia, which prevents any abrupt location changes where a delay would be noticeable.

Body and Physical Representations: Interaction in a virtual environment often requires a visual representation of the body of the viewer, particularly the hands. Additionally, other physical equipment in the interaction area may appear in the virtual environment. In the BOOM and HMD interfaces, the senses are completely restricted to the computer simulation. Here, body representation is explicit—the body must be simulated and rendered like any other kind of geometry in the environment. This requires detailed body-part position measurement and the allocation of extra rendering time.

In the CRT and CAVE environments, body representation is implicit—the body appears physically and does not require rendering. This also means the visual interface cannot alter the presence of the viewer's body. Furthermore, the body or any other physical object will occlude a virtual object even if the virtual object is closer to the viewer.

Virtual reality applications focusing specifically on body representation are the video art installations that process the image of the viewer and reproduce it in some synthesized environment. Examples are "Videoplace" [9] and E. Tannenbaum's "Recollections" video installation, on permanent display at the Exploratorium in San Francisco.

Intrusion: The intrusiveness of a virtual reality interface indicates the severity of its restriction of the senses. The HMD interface has the highest intrusion, completely isolating the viewer from the real environment. Using half-silvered mirrors allows a viewer with an HMD to see the real environment superimposed by objects in the virtual environment. Such modifications decrease the intrusion caused by the HMD, but also severely reduce the field of view. The BOOM similarly isolates the viewer, but the viewer may easily remove the interface.

The CRT and CAVE interfaces are nonintrusive. In such an environment, the viewer is free to move at will, secure in the awareness of the real, as well as the virtual, aspects of the environment.

Visualization Issues

For virtual reality to become a more useful tool, we must evaluate its visualization effectiveness. The five main visualization issues are identified in Table 2.

Visual Acuity: The quality of a computer graphics display is commonly measured by its *resolution*—the number of pixels, or individual points it uses to produce an image. The quality of a virtual reality interface is more properly measured by a combination of both resolution and field of view. This measurement is called the *visual acuity* of the display—the portion of a pixel taken from the center of the display that spans one minute (1/60th of a degree) of the field of view. Symbolically, a horizontal resolution of H pixels across a display W inches wide has a pixel pitch of $P = W/H$ inches per pixel. Given a viewer distance of D inches from the center of the display, the angle a single pixel creates on the retina is approximately $\tan^{-1} P/D$ and is measured in minutes. The visual acuity, the portion of a pixel that subtends an angle of one minute on the retina, is simply the inverse $1/\tan^{-1}(P/D)$.

The popular Snellen fraction, used to measure vision, is another unit for reporting visual acuity. A viewer whose visual acuity is indicated by the Snellen fraction $20/X$ means the viewer can see at 20 feet what a viewer with average eyesight can see at X feet. For example, a Snellen fraction of 20/20 is average, meaning a visual angle of one min-

ute is perceptible, 20/10 is above average with a minimum perceptible visual angle of one-half minute and 20/40 is below average with a minimum perceptible visual angle of two minutes. Legally blind, for example, is the accepted term for vision that cannot be corrected to better than 20/200. Residents of Illinois need a visual acuity of 20/70 or better to drive in the daytime, and at least 20/40 to drive at night. The Snellen fraction, when divided, produces the correct visual acuity as previously defined.

The following acuity measurements use the maximum (horizontal) dimension resolution whereas actual vision research tends to prefer the minimum (vertical) dimension resolution. The resolution of CRT's is considered relatively high, commonly 1,280 × 1,024 pixels. A viewer at a distance of 18 inches from a 19-inch-CRT (≈ 15″ horizontal) creates a pixel pitch of 0.0117 inches per pixel. The visual acuity of the CRT is $1/\tan^{-1}(0.0117/18) = 0.448$ pixels per minute giving the viewer 20/45 vision, which is almost good enough to drive at night.

The visual acuity of the HMD interfaces is currently limited by LCD technology. The fish-eye optics for our example, the LEEP CYBERFACE 2, complicate visual acuity computations. However, the angle subtended on the retina by a pixel from the center of the display is specified as 0.0062 radians or 21.3 minutes. This infers a visual acuity of 1/21.3 = 0.0469 pixels per minute giving the viewer about 20/425 vision, which is undoubtedly legally blind. The LEEP optics improve the poor resolution of LCD. Other HMD interfaces with an equal field of view but lacking the resolution enhancements from the LEEP optics score an even worse acuity.

The resolution of the Fake Space BOOM interface is currently about 1,000 × 1,000 (in black and white) with flexible screen widths to trade acuity for field of view. For a narrow 90° field of view, the BOOM screen width, coupled with the LEEP optics, generates center pixels subtending an angle on the retina of 0.00127 radians or 4.37 minutes. This infers a visual acuity of 1/4.37 = 0.229 pixels per minute, giving the viewer about 20/85 vision, which, except for the limited field of view, is almost good enough to drive.

Finally, the resolution of an individual CAVE screen is the same as the CRT, 1,280 pixels over 7 feet for a pixel pitch of 0.00547 feet per pixel. The viewer in the default-centered position is 3.5 feet from the center of the display. Hence, the CAVE has a visual acuity of $1/\tan^{-1}(0.00547/3.5) = 0.186$ pixels per minute, giving the viewer about 20/110 vision, which is better than legally blind but is not sufficient for even daytime driving.

Linearity: Often, the field of view and resolution of a display are increased through optical devices. These devices increase the field of view by bending the light from the displays, in effect curving a flat display around the viewer. Resolution is increased by concentrating more pixels into a small central area of the display, while leaving the edges of the display less well defined. Without such optics, the visual acuity of flat screens is worse in the center of the display and sharper at the edges—exactly the opposite of what our visual system needs.

Some manufacturers of HMD and BOOM devices use optics such as the LEEP systems [8]. These displays improve resolution at the center of the screen by as much as a factor of 2.9 and expand the field of view for each eye up to 140°.

These transformations also cause distortions that bend straight lines. The distortions are easily modeled and the inverse distortion can be computed and applied to the image to reduce the effect, though at a significant degradation in simulation speed.

Look Around: This is the ability to move about an object, viewing it from different angles. It is a useful property when using the virtual reality interface for visualization.

The CRT paradigm lacks a strong *look around* capability due to the size of its screen and its low field of view. When viewed from the side, the visible area of the CRT

Table 1.
Immersion Issues

	Field of View	Panorama	Perspective	Body Rep.	Intrusion
CRT	45°	None	Slow	Physical	None
BOOM	90° ↔ 120°	Fast	Fast	Virtual	Partial
HMD	100° ↔ 140°	Slow	Slow	Virtual	Full
CAVE	Full	Fast	Slow	Physical	None

Table 2.
Visualization Issues*

	Vis. Acuity	Linearity	Look Around	Prog. Refine.	Collab.
CRT	20/45	Linear	Limited	Fix Loc. only	Single Persp.
BOOM	20/85*	LEEP	Full	Fix Loc. and Rot.	Dup. Hardware
HMD	20/425	Either	Full	Fix Loc. and Rot.	Dup. Hardware
CAVE	20/110	Linear	Full	Fix Loc. only	Single Persp.

*At 90° field of view, black and white.

becomes much smaller, severely limiting the viewing angles from which an object is visible. One solution to this problem is to have the CRT rotate in order to always face the viewer.

The BOOM and HMD interfaces handle *look around* since they simulate everything the viewer sees. The CAVE also provides a sense of *look around.* These interfaces require some kind of virtual travel to *look around* distant objects.

Progressive Refinement: This is the ability to dynamically increase the computational expense of a model during a pause in viewer response [3]. The standard scheme simulates a fast coarse model for viewer interaction, then computes a much finer model when the viewer remains still. Coarse versus fine attributes are model resolution, such as the number of points or polygons, and rendering techniques, like adding shadows or progressive radiosity.

The HMD interface requires the viewer to remain absolutely still to refine the display. The BOOM also requires zero movement of the interface, but the high inertia and nonintrusiveness of the BOOM make this much easier than the HMD. The CRT and CAVE interfaces require only the viewer's location to remain fixed. Hence, in the CAVE in particular, the viewer is allowed to pan around during refinement.

Collaboration: One of the most important aspects of visualization is communication. For virtual reality to become an effective and complete visualization tool, it must permit more than one user in the same environment.

The BOOM and HMD interfaces allow multiple users in their environment at the high cost of duplicating the interface hardware [1]. The CRT and CAVE environments allow multiple users to benefit from the experience without modification. In such a situation, the perspective accommodates only one of the viewers. If shuttered glasses are used for stereo, then the CRT and CAVE interfaces can simulate the correct perspective for all users, though n users would see the screen only $1/n$th of the time, requiring a fast scan rate and a very bright image.

CAVE Implementation

At the time of this writing, our implementation of the CAVE uses two projection screens (two walls)—five screens (three walls, a ceiling and a floor) are expected for the Showcase '92 exhibition. The implementation of the CAVE interface requires computation of viewer-centered perspective projections, deployment of viewer tracking equipment, synchronization of displays, and overcoming any resulting projector and tracking limitations.

Viewer-Centered Perspective

The CAVE requires special perspective projections to simulate viewer-centered perspective. These projections are offset to simulate stereo, and thus require knowledge of the viewer's orientation.

Off-Axis Perspective Projections: The viewer-centered perspective, as well as the projections used for stereo, are derived from the off-axis perspective projection [7]. The simplest derivation alters a standard on-axis perspective projection by two affine transformations. First, points are *sheared* in a direction parallel to the projection plane, by an amount proportional to the point's distance from the projection plane (points in the projection plane remain unchanged). Then, points are scaled along the axis perpendicular to the projection plane by an amount again proportional to the point's distance from the projection plane (and again points in the projection plane remain unchanged). Adding stereo consists of bifurcating this projection into two similar projections differing by opposite shears along the axis of disparity—the line through the two eyes of the viewer.

The Need for Orientation: The viewer's head must be oriented. There are two reasons for this and they both involve correct stereo projection. In theatrically released three-dimensional films the viewer's head is assumed to be vertical. In the CAVE, one may want to tilt one's head. Unless the viewer's rotation about the line of sight is accounted for, one's head could tilt 180° to find an inverse stereo effect, or 90° to find no stereo effect at all. These concerns become paramount when considering stereo projection onto the floor and ceiling.

The position of the viewer's eyes is needed to prevent inconsistencies at the edges of the CAVE walls. If stereo is computed assuming the user is looking perpendicular to the

One of the most important aspects of visualization is communication. For virtual reality to become an effective and complete visualization tool, it must permit more than one user in the same environment.

projection planes, the stereo disparities will not line up at the edges where two projection planes meet.

Display Hardware

Real-time rendering of the virtual world is achieved through six Silicon Graphics Inc. VGX workstations, each attached to a rear projection display. A Silicon Graphics "Personal Iris" serves as a master controller for the system and all workstations communicate via Ethernet.

Multiple Stereo Displays: These workstations display stereo, using the StereoGraphics "CrystalEyes." StereoGraphics divides the VGX frame buffer into two half-vertical-resolution fields, one for each eye. The user wears liquid crystal glasses that shutter at the field rate of the displays, synchronized by an infrared signal. At a rate of 60Hz (30Hz per eye) the display flickered noticeably at highly disparate areas. Hence, the update rate was doubled to 120Hz.

Multiscreen stereo requires synchronizing the video signals. The 120Hz screen update rate produced by the StereoGraphics hardware was not compatible with the VGX genlock input. The only prototype hardware unit used in the system, a stereo sync processor, fixed this problem. This processor filters out every other sync signal emitted by the source VGX, creating a 60Hz signal that the slave VGX genlock inputs could handle.

The Green Problem: An unexpected side effect of the 120Hz video rate was a lag in the green channel of the video. This is due to the long decay time of the green phosphors in the video projectors. The display would alternate from eye to eye faster than the green phosphors could handle. The resulting double image in the green channel caused a complete loss of stereo depth perception.

One immediate solution was to work in the hue space spanned by the red and blue axes which reduced brightness to 41%. Another solution was to fix green at some level, say 50% which increased the brightness and allowed the full hue space at the expense of one brightness per hue and reduced contrast. We are now using faster green phosphors that completely solve the green problem.

Viewpoint Tracking: The position and orientation of the user's head is obtained with a 3SPACE Polhemus "Isotrack" sensor, whose transmitter is mounted on the StereoGraphics glasses. As expected, there is a noticeable sensing lag, which manifests itself during fast viewer motions. As stated earlier, this is not a problem for viewer rotation, but remains a problem when moving. Extrapolation techniques are currently being investigated to better predict user motion to reduce the interactive delays due to sensor lag.

Conclusion

The CAVE is a nonintrusive easy-to-learn high-resolution virtual reality interface. It is superior to other virtual reality paradigms across many issues, particularly in field-of-view, visual acuity and lack of intrusion. Moreover, it is open to limited use for collaborative visualization.

Applications: SIGGRAPH '92 Showcase

Several applications of the CAVE will be featured at Showcase. These include applications from the Electronic Visualization Laboratory, and others.

Regional-Scale Weather in Three Dimensions: This application is from the work of A. Campbell at the Argonne National Laboratory. It uses the PSU/NCAR Mesoscale Model in a parallelized form, running interactively on the Intel Touchstone Delta, to create a three-dimensional display of weather systems over a region of North America.

Graphical Planning for Brain Surgery: Brain-surgery-planning software, featured by R. Grzeszczuk, is currently undergoing clinical testing at the University of Chicago. It employs a three-dimensional localizer as means of interactively transferring spatial relationships from MR-derived three-dimensional anatomical models directly onto the patients.

The Visible Embryo: The viewer is taken on a trip through a human fetus via a simulation developed by L. Sadler and the Biomedical Visualization Laboratory at the University of Illinois at Chicago, providing a unique view of the human body that could aid in medical developments.

The Snowstorm: This project visualizes three-dimensional vector fields using interactive particle systems where small points traverse the vector field, each at speeds proportional to the magnitude at that point in the field. Predefined vector fields are provided as well as the ability to "comb" new vector fields interactively using a wand.

Fractal Exploratorium: A virtual laboratory of fractals and chaotic attractors is presented in this application by R. Hudson of the Electronic Visualization Laboratory. Participants can investigate chaotic forms from a variety of different perspectives, interactively altering their parameters, and hence, their shapes.

Bio Modeling: The interactive modeling of biological macromolecules is demonstrated in this application from K. Shulten of the Beckman Institute at the University of Illinois at Urbana-Champaign.

The Evolving Universe of Galaxies and Stars: A combination of stored database images and real-time computations from a remote CRAY will allow the viewer to fly through an evolving universe in this application developed by M. Norman at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign.

Further Research

Most of the problems with the CAVE are a consequence of hardware shortcomings, such as lag time due to tracking delays, the green

problem due to insufficient phosphor decay times, and multiprojector stereo synchronization. Moreover, brighter screens with faster update rates would allow multiple-viewer-centered perspective.

The effectiveness of virtual reality interfaces, particularly the CAVE, need to be evaluated more quantitatively. The degree of immersion an interface creates as well as its ability as a visualization tool are difficult quantities to obtain and deserve a much more thorough treatment than given here.

Acknowledgments

We would like to thank Gary Lindahl and Robert Grzeszczuk for technical support and assistance. Graduate art students Michelle Miller, Kathy Koller and Lewis Siegel created the system diagrams.

Gary Oberbrunner of Thinking Machines Inc. simultaneously suggested the extrapolation technique. Thanks to Sumit Das for an implementation of this method.

The visual acuity and linearity issues used information graciously provided by Eric Howlett at LEEP Systems, Inc., for which we are grateful. We would also like to thank the Brenart Eye Care Center of Yorkville, Ill. for its assistance with the eye chart diagrams and vision assessment.

This publication, and the research and conclusions made thereby, were supported in part by a grant awarded by the Illinois Department of Commerce and Community Affairs. Representations made by this publication do not necessarily reflect the opinions and conclusions of the department.

References

1. Blanchard, C., Burgess, S., Harvill, Y., Lanier, J., Lasko, A., Oberman, M. and Teitel, M. Reality built for two: A virtual reality tool. *Comput. Graph. 24, 2 (Mar. 1990), 35–36.*
2. Brooks, F.P. Grasping reality through illusion: Interactive graphics serving science. In *Proceedings of SIGCHI '88* (May 1988), pp. 1–11.
3. Brooks, F.P., Ouh-Young, M., Batter, J.J. and Kilpatrick, P.J. Project GROPE: Haptic displays for scientific visualization. *Comput. Graph. 24,* 4 (Aug. 1990), 177–185.
4. Chung, J.C., Harris, M.R., Brooks, F.P., Fuchs, H., Kelley, M.T., Huges, J., Ouh-Young, M., Cheung, C., Holloway, R.L. and Pique, M. Exploring virtual worlds with head-mounted displays. In *Proceedings of SPIE,* Vol. 1083, Feb. 1990, pp. 42–52.
5. Fisher, S. The AMES Virtual Environment Workstation (VIEW). *SIGGRAPH '89* Course #29 Notes, Aug. 1989.
6. Fisher, S. Viewpoint dependent imaging: An interactive stereoscopic display. In *Proceedings of SPIE,* Vol. 367, Feb., 1982.
7. Hodges, L.F. Time multiplexed stereoscopic computer graphics. *IEEE Comput. Graph. Appli. 12,* 2 (Mar. 1992), 20–30.
8. Howlett, E.M. Wide angle orthostereo. In *Proceedings of SPIE,* Vol. 1256, Feb. 1990, pp. 210–223.
9. Krueger, M.W. *Artificial Reality II.* Addison-Wesley, 1991.
10. Lippman, A. Movie Maps: An application of the optical videodisc to computer graphics. *Comput. Graph. 14,* 3 (1980).
11. Pausch, R. Virtual reality on five dollars a day. In *Proceedings of SIGCHI '91,* 1991, pp. 265–270.
12. Plato. *The Republic.* The Academy, Athens, circa 375 BC.
13. Rheingold, H. *Virtual Reality.* Summit, N.Y., 1991.
14. Rolfe, J.M. and Staples, K.J. *Flight Simulation.* Cambridge Univ., 1986.
15. Schachter, B.J. Computer image generation systems. In *Computer Image Generation,* B.J. Schachter, Ed., Wiley & Sons, N.Y., 1983.
16. Sutherland, I.E. The ultimate display. In *Proceedings of IFIP 65,* 2, pp. 506–508, 582–583.
17. Teitel, M.A. The Eyephone: A head-mounted stereo display. In *Proceedings of SPIE,* Vol. 1256, Feb. 1990, pp. 168–171.
18. Venolia, D. and Williams, L. Virtual Integral Holography. Tech. Rep. 90-10, Apple Computer Inc., Feb. 1990.

CR Categories and Subject Descriptors: I.3.2 [**Computer Graphics**]: Graphics Systems—*Distributed/network graphics;* I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Virtual reality*

General Terms: Human Factors

About the Authors:

CAROLINA CRUZ-NEIRA is a Ph.D student at the Electronic Visualization Laboratory. Her current research interest is in interactive computer graphics and virtual reality.

DANIEL J. SANDIN is codirector of the Electronic Visualization Laboratory and a professor in the School of Art and Design at the University of Illinois at Chicago. His current research interests are in interactive computer graphics and rendering of diffraction. email: sandin@siggraph.org

THOMAS A. DEFANTI is codirector of the Electronic Visualization Laboratory, professor in the department of electrical engineering and computer science, and director of the UIC Software Technologies Research Center. His current research interests are in interactive systems and computer graphics languages. email: tom@siggraph.org

ROBERT V. KENYON is an associate professor in electrical engineering and computer science at the University of Illinois at Chicago. His research interests include sensory-motor adaptation, visuomotor control, effects of microgravity on vestibular development, flight simulation, computer graphics, and digital signal processing using parallel computers. He is currently working on problems of modeling complex visual/motor coordination in the human. email: kenyon@uicbert.eecs.uic.edu

JOHN C. HART holds a joint postdoctoral research associate position at the Electronic Visualization Laboratory, University of Illinois at Chicago and the National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign. His current research interests are in the modeling and rendering of fractals. email: hart@uicbert.eecs.uic.edu

Authors' Present Address: Electronic Visualization Lab., EECS Dept. M/C 154, Univ. of Ill., Chicago, IL. 60680-4348; fax: 413-7585

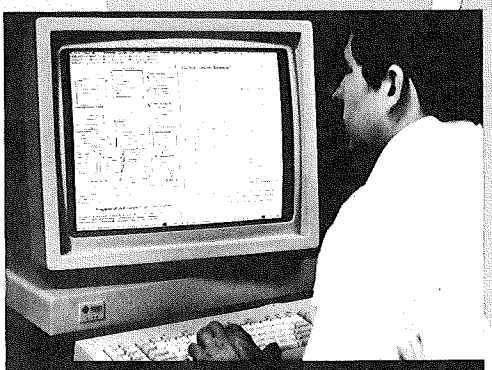
Announcing the primary source of information on the hypertext revolution

HYPERTEXT Compendium

With more than 115 articles, reports, speeches and studies by the people who have virtually invented modern hypertext technology and applications, this is the most complete collection of material on hypertext technology...in a fully-integrated hypertext format, extensively interlinked and cross-referenced.

Hypertext is a way of using information that allows you to make the computer "think" a lot more like you do at your most creative. The *ACM Hypertext Compendium* enables you to explore hypertext principles and concepts by using hypertext technology itself to conduct a wide-ranging or carefully-targeted search through the most authoritative information available .

You can use the *ACM Hypertext Compendium* in any of three versions; see the order form below.



WHAT'S IN IT FOR YOU?

COMPUTING PROFESSIONALS:
Build a system with a hypertext front end. Learn innovative ways to connect and manage all types of applications--from data processing to end-user query systems.

PROFESSORS AND STUDENTS:
Use the single best information source in the field, covering everything from managing links, to developing underlying algorithms, to solving storage, retrieval and interface problems.

HYPERTEXT SPECIALISTS:
Refresh your memory on Halasz's overview of Notecards. Prepare your next paper or review Hypertext '90 panel discussions with this primary research tool.

APPLICATIONS DEVELOPERS:
Increase efficiency, quality and profitability. Innovative uses for hypertext documents are virtually unlimited.

QUESTIONS?
Contact Bernard Rous or Margaret Tuttle at ACM: 212-869-7440.

Please fill out this order form and send with your payment to:
ACM Order Dept., P.O. Box 64145, Baltimore, MD 21264
or call: 1-800-342-6626 (AK, MD, outside U.S. 301-528-4261)

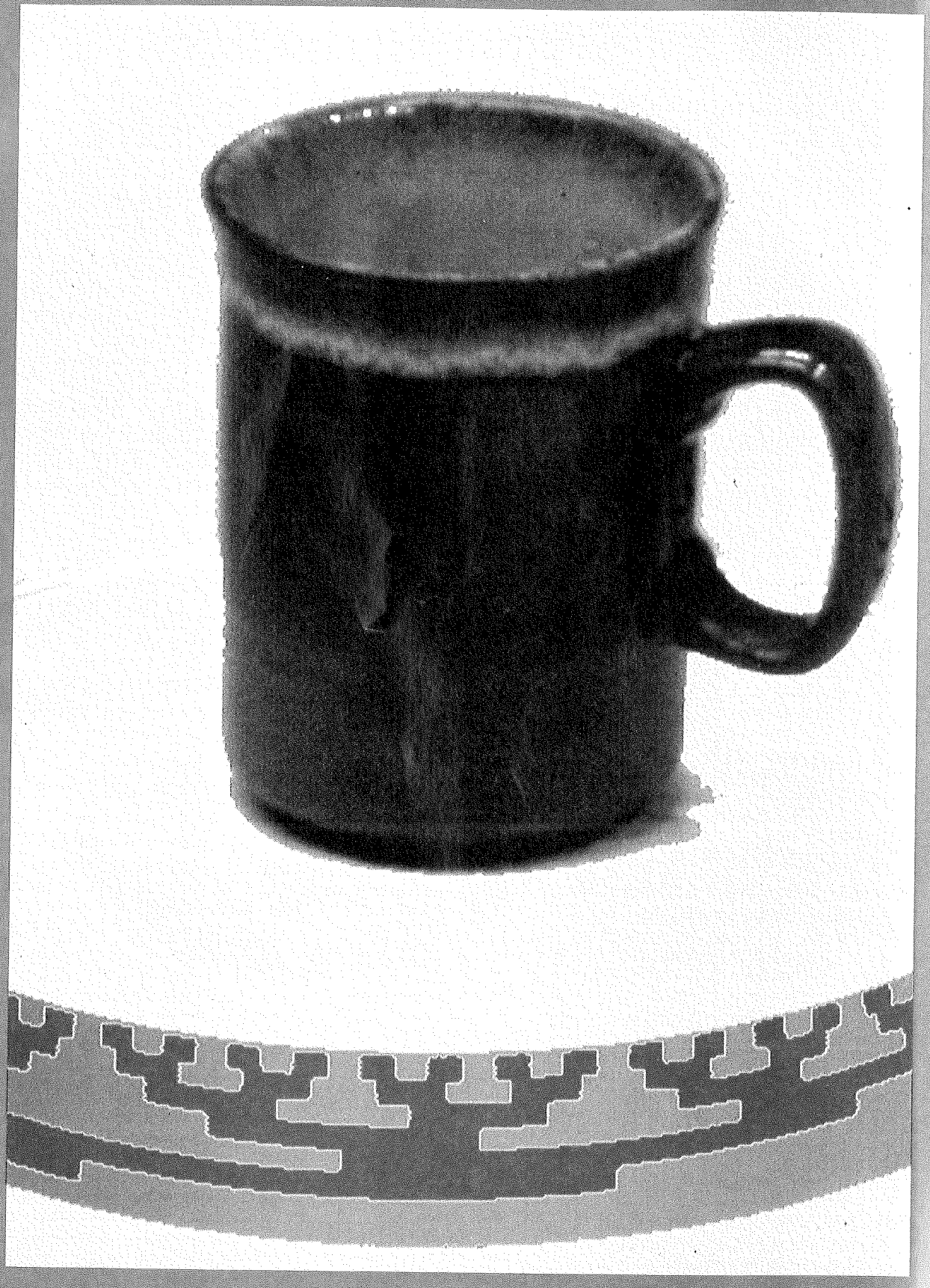
Check version:	ACM Order #	ACM Member Price	Nonmember Price
___ **KMS Hypertext Software Version** (from Knowledge Systems, Inc.). Requires Sun 3 or 4 Workstation.	217910	$295	$495
___ **Hypercard Version.** Requires Hypercard 2.0 or 2.1. If 2.1 is run under System 7, 2MB RAM required. Application has been optimized for display on 13" or larger monitor. Use on 12" or smaller monitor requires frequent vertical/horizontal scrolling.	217913	$125	$199
___ **ASCII Version.** (Provides articles only with links marked; no graphics)			
Please check: ___ MS DOS disks	217915	$100	$179
___ Macintosh disks	217914	$100	$179

Shipping on orders totaling: $100.00 (+ $9); $100.01 to $500.00 (+ $10); $500.01 to $1000.00 (+ $13); $1000.01 and over (+ $19)

Subtotal _____
Shipping _____
Total _____

Name_____Co./Inst. _____
Street Address_____
City, State/Province, Post. Code_____
Telephone # (for questions about your order) _____
___ I am an ACM member. Member #: _____(Required for Member Price)
___ Payment enclosed, in the amount of $ _____ (Check payable to ACM, Inc.)
Or: *(Payment must be in U.S. funds drawn on a U.S. bank.)*
___ Please charge ___ Visa ___ MasterCard ___ American Express
Account # _____Exp. date _____ Signature_____
(must be signed)

711



Modeling and Analysis of Empirical Data in Collaborative Environments

Ingrid Carlbom
William M Hsu
Gudrun Klinker
Richard Szeliski
Keith Waters
Michael Doyle
Jim Gettys
Kristen M. Harris
Thomas M. Levergood
Ricky Palmer
Larry Palmer
Marc Picart
Demetri Terzopoulos
David Tonnesen
Michael Vannier
Greg Wallace

Many tasks in such diverse fields as medicine, oil exploration, and mechanical design depend on the collection and interpretation of empirical data. Our long-term vision is a collaborative scientific visualization environment, where scientists, engineers, and physicians work together on modeling and analyzing empirical data using an integrated set of tools and techniques from computer graphics, computer vision, and image processing. This environment would also take advantage of interaction techniques that include sound, spoken commands, and gestures.

To illustrate how such an environment could be used, let us look at some hypothetical scenarios. In the first scenario, a radiologist gives an "interactive report" on findings in a CT scan to a surgeon at a remote location. This report contains three-dimensional data that is manipulated by the radiologist to illustrate features of interest. The surgeon can see the radiologist as the report is delivered. The surgeon can also interject questions and manipulate the data to clarify the questions. Alternatively, this report can be left in the surgeon's electronic mail, to be accessed at a later time. Similarly, the surgeon may want to consult the radiologist from the operating room and discuss a preoperative or intraoperative scan. The surgeon and the radiologist can carry on a conversation between the operating room and the radiologist's office, while the radiologist monitors the operation on a workstation. In addition, both doctors can simultaneously manipulate, point at, and analyze the three-dimensional data.

A second scenario from the field of medicine is preoperative facial analysis and postoperative quantitative evaluation. Patients expecting surgery to improve their facial paralysis are preoperatively scanned to construct a synthetic facial model. In this model, synthetic muscles are activated to simulate and improve the patient's appearance. After the surgery, the surgeon and the referring neurologist can examine any improvement by quantitatively comparing pre- and postoperative image sequences of the patient making facial expressions.

In many research domains, the ability to share massive amounts of empirical data is crucial to significant progress. One example is neuroscience, where researchers work only on a very small part of the bigger problem of mapping the brain and its functions. Published papers may contain only one or two images of a 2,000-image dataset, which is in general inaccessible to other researchers. Currently, there is a nationwide effort to establish a central database of neuroscience data [11], and a collaborative visualization environment could be used to access such a database.

In another domain, engineers at different locations collaborate on the design of a new product. Starting with a geometric model of last year's version, they interactively modify and refine the model through local changes to its shape, color, and texture. Alternatively, a manufactured object, such as a competitor's product or a clay mock-up, may be the starting point from which a computer-aided design (CAD) model is automatically created and subsequently modified. As in the biomedical examples, the engineers can consult colleagues and collaboratively modify the design over the computer network.

To support such scenarios, we envision a scientific visualization environment incorporating video, sound, and user interfaces that understand spoken requests and gestures. This vision suggests a seamless environment in which the user can consult a colleague, while the conversation and respective faces are transmitted and displayed. Furthermore, the computer is no longer passive, but an active participant understanding limited speech, responding with an expressive face and a synthesized voice.

The sources of data in the preceding examples, namely CT scanners, transmission electron microscopes, and video cameras, are different and the different types of data require considerable domain knowledge in their analysis. Nevertheless, there is a great deal of commonality in the techniques used to visualize, quantify, and interact with the data and the models. These common techniques range from data enhancement, feature extraction, and reconstruction, to interactive visualization, computation, and simulation on the models.

We are capitalizing on these similarities by building a common scientific visualization environment which supports such diverse applications as biomedical scientific visualization and reverse engineering applications. We discuss three projects we are pursuing toward our goal of such an environment: (1) interactive modeling and visualization of medical and biological data, (2) three-dimensional shape acquisition, modeling, manipulation, and culminating in three-dimensional faxes, (3) teleconferencing with personable computers. These projects are demonstrated at the SIGGRAPH'92 Showcase.

Interactive Modeling and Visualization of Medical and Biological Data

Interactive modeling and visualization of medical and biological data in a collaborative environment has the potential for improving patient care and reducing medical costs. Visualization environments for these domains must address all stages of data analysis, including registration, segmentation, three-dimensional reconstruction, rendering, analysis, and simulation. We discuss some fundamental methods in each area and demonstrate examples from neuroscience, embryology, radiology, and surgical planning. The components of the visualization environment are implemented in a collaborative environment, suggesting that biomedical imaging, medical diagnosis, and surgical planning will soon be feasible over high-speed networks, allowing electronic intra- and inter-hospital collaboration among physicians and researchers.

Registration

Registration refers to the alignment of data from the same or different modalities, or sensors, such as the alignment of slices of neural tissue, or the alignment of MR and CT data. We use two registration techniques: interactive registration with a digital blink comparator, and automatic registration through minimization. The first, the digital blink comparator, is a manual technique using visual motion for the alignment of images. Holding one image stationary, the user translates and rotates the other image while the stationary and moving images are alternately shown on a graphics screen. The images are aligned when the sum of the visual motion is minimized [5]. The blink comparator is used to register sections of a neuronal dendrite from transmission electron microscopy, as well as pre- and postcontrast midsaggital MR head scans.

The automatic method attempts to minimize the difference between the two images to be registered. One image is kept stationary and the other is rotated and translated, in a systematic manner, with respect to the stationary image. The transformation parameters which yield the minimum difference between the images also yield the optimum alignment. This registration technique is used to register serial sections of an embryo from light microscopy.

Segmentation and Reconstruction

Segmentation refers to the process

of extracting meaningful regions from images or volumes. We employ a user-assisted segmentation method, *snakes,* or interactive deformable contours, which is a model-based image feature localization and tracking technique [5,7]. Although consistent with manual tracing methods, snakes are considerably faster and more powerful. The user quickly traces a contour which approximates the desired boundary, then starts a dynamic simulation that enables the contour to locate and conform to the true boundary. Where necessary, the user may guide the contour using a mouse to apply simulated forces. With some guidance, snakes exploit the coherence between adjoining images to quickly extract a sequence of regions. Snakes are used to extract a neuronal dendrite from transmission electron micrographs (Figure 1), and an embryo heart and its substructures from light micrographs, and to track lung and liver vessels in spiral CT scans. After extracting all the regions of an object, we stack them to form three-dimensional volumetric models of a neuronal dendrite (Figure 2a), an embryo heart, and the heart and liver vessels.

Rendering

The three-dimensional volume data is rendered using a novel parallel volume ray-tracing algorithm. This ray-tracing algorithm differs from previous methods by holding the data stationary while accumulating the opacity along the rays in parallel. The three-dimensional volumetric models can be interactively rotated, cut open (Figure 2b), and viewed transparently. The three-dimensional shapes of the models are displayed, using both shading and motion parallax. Emphasis is placed on data fidelity, or *loss-less* rendering [4], using accurate interpolation filters [3].

Simulation

In a different application, we simulate facial tissue dynamics for surgery planning. Two iso-valued surfaces of the facial skeletal bone and epidermal skin tissue are extracted from CT data using the *marching cubes* algorithm [10] (Figure 3). These geometric surfaces provide a foundation for a discrete layered spring lattice tissue model [21]. Three layers are constructed for the numerical simulation: the skeletal bone, the muscle, and the epidermis. One end of each synthetic muscle attaches to the muscle layer and the other end attaches rigidly to bone. When a muscle is articulated, it deforms the lattice, which causes forces to propagate outward until an equilibrium is established. The simulation is implemented on a massively parallel computer to achieve a rapid response.

Three-Dimensional Shape Acquisition, Modeling, and Manipulation

Three-dimensional modeling is required in applications such as computer-aided design and manufacturing (CAD/CAM), architectural design, and computer animation. Traditionally, three-dimensional model entry involves a laborious manual process based on two-dimensional input devices such as tablets. The recent advent of interactive three-dimensional input techniques [12] and automatic shape acquisition using special-purpose three-dimensional active rangefinders [6] could alleviate some of these problems. However, these approaches rely on expensive special-purpose input devices that may not be available to the general public.

In this section we describe an alternative shape acquisition technique based on regular video images, which we expect to become widely applicable as video technology becomes embedded in workstations. In our system, the object to be scanned rotates on a turntable in front of an ordinary, stationary video camera (Figure 4a). We have developed two algorithms to reconstruct the object from the resulting image sequence: shape from silhouettes, and shape from image flow. On acquiring the object's shape, we can interactively modify it using a variety of three-dimensional manipulation techniques.

Three-Dimensional Shape from Silhouettes

Our first algorithm for shape acquisition constructs a bounding volume for the object from the sequence of *silhouettes* [15]—the binary classification of images into object and background (Figure 4b). The three-dimensional intersection of these silhouettes defines a bounding volume within which the object must lie. We represent this volume using an octree [13] (Figure 4c). For each silhouette, the octree cubes are projected into the image plane and classified as either wholly or partially inside or outside the object. After one complete revolution at a given resolution, cubes which cannot be unambiguously classified are subdivided into eight subcubes, and the process is repeated [15]. Once the complete shape of the object has been reconstructed, we associate each octree cube with a set of pixels in the input images, producing a "texture-mapped" three-dimensional object. The complete procedure can be performed in a few minutes on a workstation, in time proportional to the desired resolution.

Three-Dimensional Shape from Image Flow

Our second algorithm computes the *optic flow* (two-dimensional motion) at each pixel to estimate the three-dimensional location of points on the surface of the object [16]. The optic flow is computed from successive pairs of images by minimizing a correlation measure at each pixel. This produces both a dense estimate of the local motion (Figure 5b), and a confidence measure which depends on the amount of variation in the local texture in the image (Figure 5c). Flow measurements are converted into three-dimensional points on the surface using the known turntable motion (Figure 5d). These three-

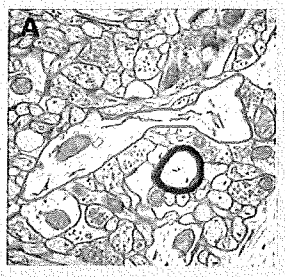
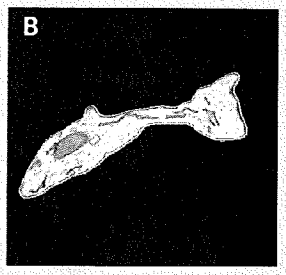
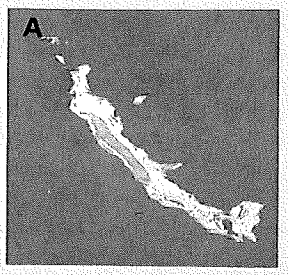


Figure 1.





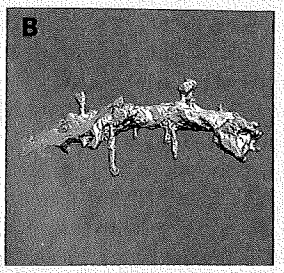
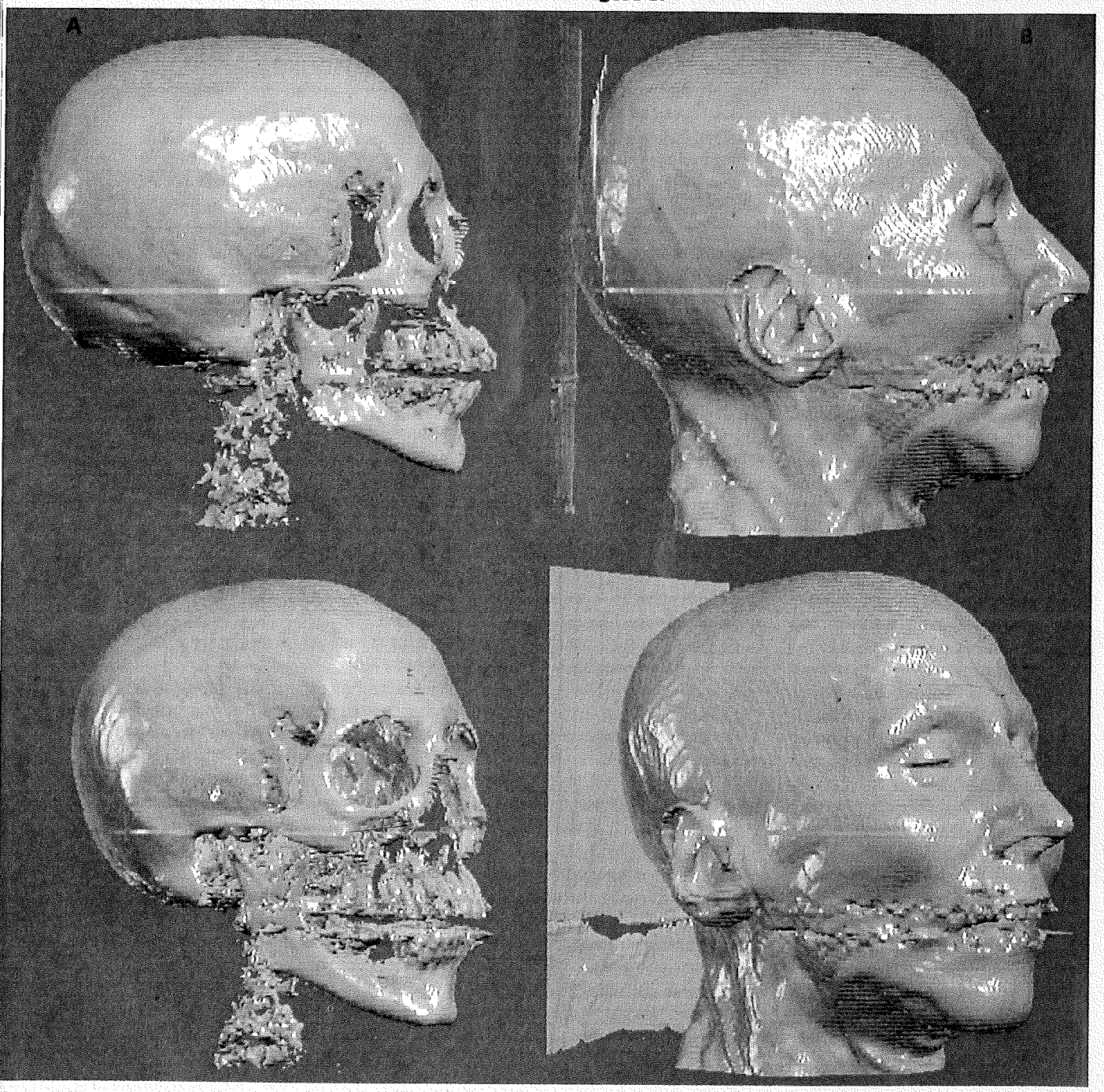


Figure 2.

Figure 3.



dimensional positions are then refined by merging them with new measurements from successive images using a statistical framework [16]. The final model is a collection of points on the object's surface tagged with colors and intensities derived from the set of images. Since the computation of flow and the merging points are computationally intensive, we have implemented these algorithms on a massively parallel processor.

Three-Dimensional Surface Interpolation and Manipulation

Because the flow-based reconstruction algorithm only produces estimates at locations with sufficient texture, there may be gaps in the surface. To solve this problem, we have developed a flexible surface modeling technique based on interacting particles [17]. Our method interpolates across gaps in the surface by placing particles at the initial surface measurements and then adding new particles automatically. The surface is automatically smoothed, based on specially tailored interaction potentials. Each particle is then colored with an appropriate intensity derived from the initial image sequence, producing a texture-mapped surface model of the object. To refine or reshape the object, we can use traditional techniques such as global or local free-form deformations [2,14], as well as our particle-based surface modeler. This latter approach is capable of extending or topologically altering surfaces by cutting or merging particle sheets [17].

Applications

The primary application we demonstrate at the SIGGRAPH'92 Showcase is a three-dimensional fax with collaborative design revision. By three-dimensional fax, we mean the ability to transmit to a remote site the full three-dimensional description of a real object. Once the

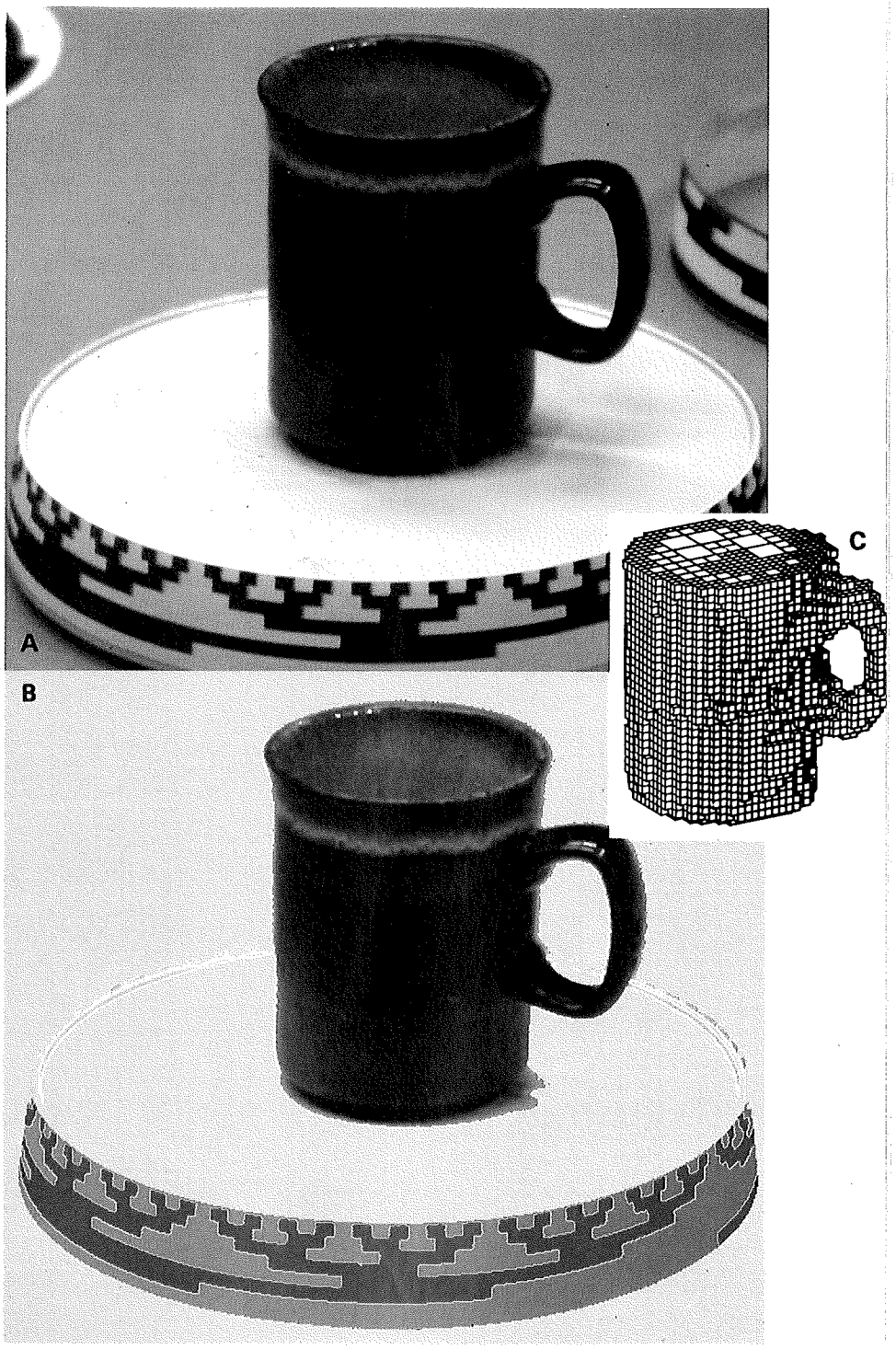


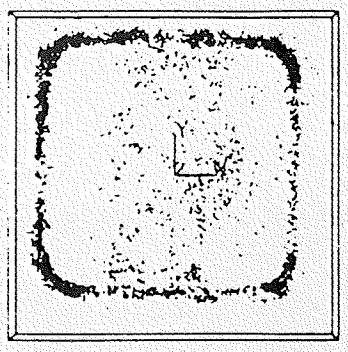
Figure 1. Cell segmentation using a deformable contour a) dendritic profile b) segmented cell

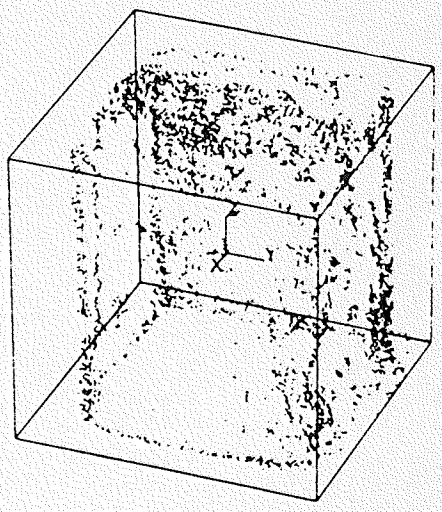
Figure 2. This reconstructed dendrite was built using an earlier version of our system which was implemented on a Silicon Graphics 4D/220GTX. VoxelView™ was used for volume rendering.

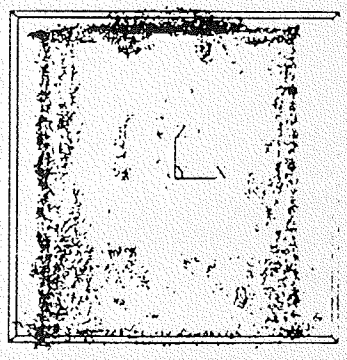
Figure 3. a) Iso-value surface of bone b) Iso-value surface of skin tissue

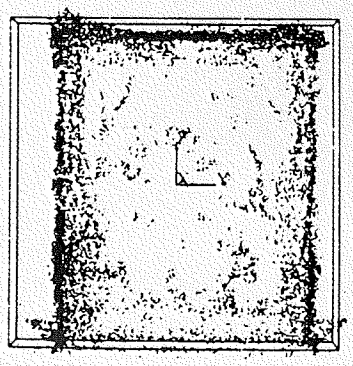
Figure 4. Shape reconstruction from image silhouettes: a) one of the original images, b) a thresholded silhouette image and c) an octree representation of the internal model

D

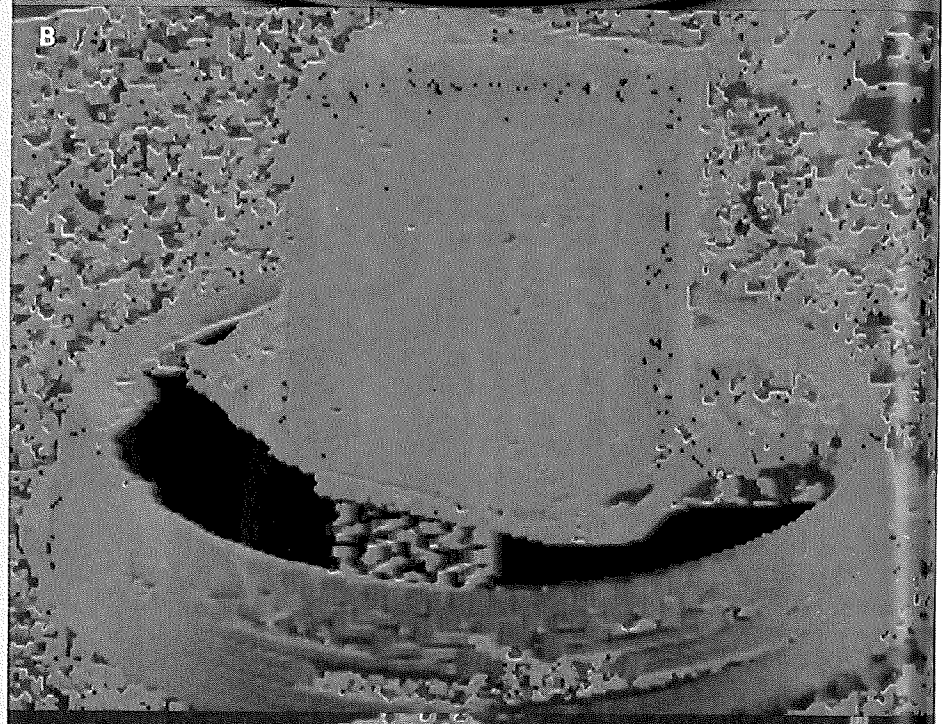




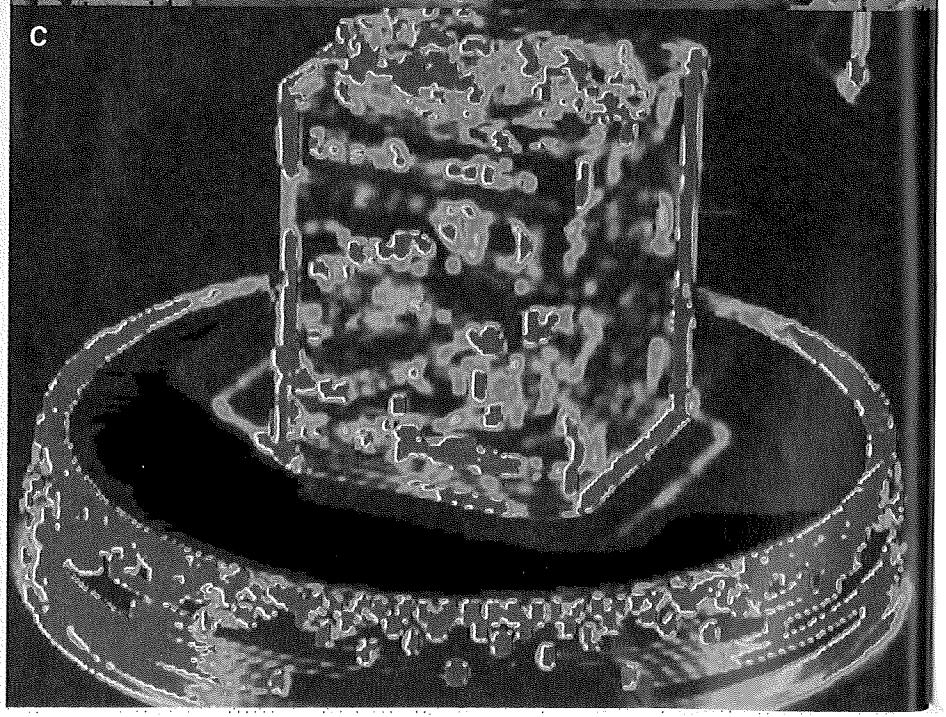




A



B



C

three-dimensional object model has been entered, it can be interactively modified or refined at each site, both through local changes to its shape, and through manipulation of the texture map (painting). The object could also be reproduced in three-dimensions using NC milling or stereolithography.

These techniques also have applications in reverse engineering (deriving CAD descriptions from manufactured objects), in the creation of virtual reality environments, and in the creation of object models for computer graphics animation. We expect the addition of simple automatic shape acquisition capabilities, as well as the wider availability of networked display and manipulation capabilities, to greatly enhance the usefulness and power of existing three-dimensional modeling systems.

Teleconferencing with Personable Computers

The face is a powerful medium of communication, and humans are highly tuned to comprehend subtle and complex facial signals. An articulate synthetic face suggests novel scenarios for presenting information. By adding such a face to synthetic speech, we can increase the bandwidth and the expressiveness of the spoken word. If the synthetic speech/face generator were combined with a system that performs basic facial analysis of the user, tracks the focus, decodes emotional states from facial expression, and analyzes the user's speech, we would transform the computer from an inert box into a *personable computer.*

How does this technology fit into a collaborative scientific visualization environment? Perhaps, it could be the front end of an expert system for empirical data analysis. A synthetic character could present some technical knowledge and, on being asked a question, explain how a piece of information was derived from the existing data. The advantage of the synthetic character over a textual interface will, we conjecture, be similar to the advantage of interacting with a person rather than reading explanations to understand a technical problem.

Within the bounds of today's technology we can demonstrate teleconferencing of today and of the future. Users can talk not only to people but also to a remote computer answering simple questions. The computer will have a personable character with expressive faces synchronized with synthetic speech.

Modeling Three-Dimensional Facial Structure

Three-dimensional face data captured from high-speed active scanners can be used to model facial structure [6,19]. These datasets are mapped onto a discrete deformable mesh with a known topology, enabling the use of a standard articulation model, and also reducing the size of the data. This deformable mesh behaves as an elastic mask, with some parts fixed on key areas of the face (e.g., eyes), and others free to move over the facial structure. Once the mesh establishes an equilibrium, it is frozen and used as the foundation for articulation (Figure 6).

Articulation and Control

We have developed techniques for articulating facial geometries through the use of synthetic muscle actuators [20]. This approach, augmented by facial tissue models, yields realistic facial expressions [21]. Two primary muscle types are used in the facial model: linear muscles which pull in an angular direction, and sphincter muscles which squeeze, like the drawing together of a string bag. Only the most significant facial muscles are used. These muscles can operate in isolation or as small functional groups to generate facial expressions and articulate the mouth for speech. Facial expressions such as happiness, sadness, anger, fear, disgust, and surprise are accomplished by grouped muscle activities (Figure 7).

Speech Synchronization

The facial model is synchronized to an automated speech synthesizer [8]. The synthesizer converts regular text into a phonetic transcription annotated with timing, intonation, and stress information as well as audible sound. The phonetic transcription is coordinated with the muscle activation of the lips, resulting in a synthetic character that appears to speak.

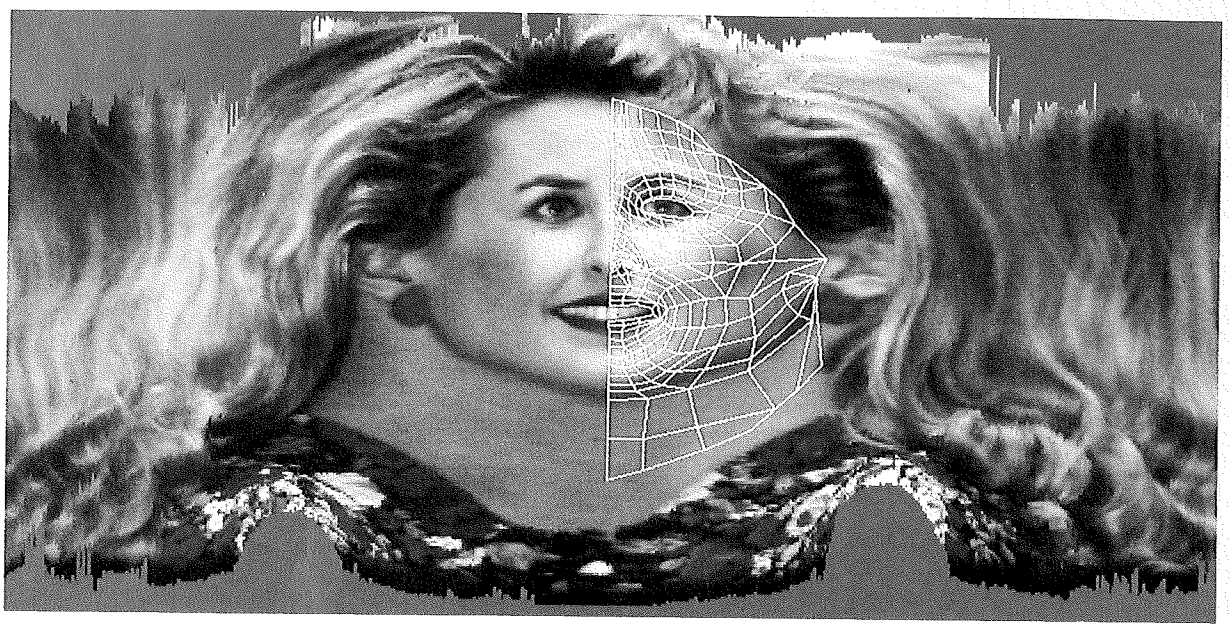
Demonstrations

We demonstrate two scenarios at the SIGGRAPH'92 Showcase. First, video teleconferencing of today, using color images of real people interacting and talking while their mutual images are displayed in live video windows. Second, teleconferencing of tomorrow, using synthetically generated facial images. The user is able to communicate with a remote personable computer and obtain responses to a limited set of questions.

Hardware and Software

The primary platform for the demonstrations is the DECmmp 12000, a massively parallel processor, with several DECstations networked over FDDI. The bidirectional person-to-person video teleconferencing uses an experimental video JPEG compression/decompression board which provides real-time compression/decompression of video into the DECstation's main memory, as well as audio capability using DECaudio. The video approach is unusual in that video is treated as a normal data type by using unextended X servers for display rather than back-door paths into the frame buffers, as has generally been the case with "video in a window" in the past. For our inter-

Figure 5. Shape reconstruction from image flow: a) one of the original input images, b) local flow estimates (blue to the right, red to the left), c) certainty in flow estimates (red is highest certainty), and d) final 3D point cloud computed from all views.



active visualization and modeling, we use a version of AVS [18] running on top of shared X [1], extended with special-purpose modules for registration, segmentation, model building, and rendering.

Future Research

Based on the demonstrations described and other experiences from our research, we have identified a number of basic capabilities which must be added to current scientific visualization environments to make them more applicable to a wide range of data analysis problems (Also see [4,9]):

- Three-dimensional modeling and visualization environments must support a wide range of representations. For example, in the three-dimensional object reconstruction system, both octrees and distributed surface representations based on particles have been used, and additional representations based on more traditional CSG primitives and NURBS could be added. In the medical applications, two-dimensional images, two-dimensional and three-dimensional contour and surface models, and three-dimensional volumetric representations are all important.
- A large number of data presentation modalities must be available, including images, height fields, contours, surfaces with texture mapping, and volumes. The data presentation should *not* be determined by the internal data representation. For example, while an image is usually displayed as a gray-level or color image, it could also be displayed as a height field or as an array of numbers. Similarly, a volume can either be ray-traced or displayed as a sequence of images using visual motion to convey the internal shape.

Figure 6. Face lattice template on the range map

Figure 7. Facial expressions created by muscle articulation

- The scientific visualization environment should support concurrent analysis of multiple datasets from the same or different modalities and from the same or different subjects. The user should be able to visually correlate parts of these datasets in many different ways, through overlays or pointing in multiple windows.
- The rapid display and three-dimensional manipulation of data facilitates the use of motion parallax to discern three-dimensional shape. Accurate, or loss-less, rendering is also important, as is the quantification of any loss of accuracy in the data due to rendering.
- The modeling and visualization system should include automatic and semiautomatic image and data reconstruction techniques. We expect such capabilities, which do not exist in most current visualization systems, to play an increasingly important role. Existing registration and segmentation techniques do not perform well on broad classes of data. A wide variety of special-purpose registration and segmentation techniques must be developed and integrated into the environment, while research into more generally applicable techniques must continue. Similarly, the recovery of three-dimensional shape and color distributions is becoming possible under controlled conditions, and provides a powerful tool both for model acquisition and for more general image analysis. The solution of the general problems of shape recovery and segmentation is difficult, however, and is the focus of much research in computer vision.

Finally, many problems in scientific visualization are complex and require the cooperation of many different experts. In the future, these will include computers as active collaborators. A collaborative modeling and visualization environment would facilitate this task, provided that we can extend interaction paradigms for complex data analysis to multiple collaborating users.

While many of these components involve open research questions, we believe that rapid progress is being made. Including a richer set of existing analysis, modeling, and presentation techniques should greatly enhance the power and usefulness of today's modeling and visualization environments. In the future, we expect continued advances in these components, as well as increases in the performance of computers and communication networks, resulting in a new generation of visualization environments that will have a profound impact on collaborative work. ■

References

1. Altenhofen, M., Neidecker-Lutz, B., and Tallett, P. Upgrading a window system for tutoring functions. *European X Window System Conference and Exhibition (EX '90)* (Nov. 1990).
2. Barr, A.H. Global and local deformations of solid primitives. *Comput. Graph. (SIGGRAPH'84), 18,* 3 (July 1984), 21–30.
3. Carlbom, I., Chakravarty, I., and M Hsu, W. SIGGRAPH'91 workshop report. Integrating computer graphics, computer vision, and image processing in scientific applications. *Comput. Graph. 26,* 1 (Jan. 1992), 8–17.
4. Carlbom, I., Terzopoulos, D., and Harris, K.M. Reconstructing and visualizing models of neuronal dendrites. In *Scientific Visualization of Physical Phenomena,* N.M. Patrikalakis, Ed., Springer-Verlag, N.Y., 1991, pp. 623–638.
5. Cyberware Laboratory Inc. *4020/RGB 3D Scanner with color digitizer.* Monterey, 1990.
6. Kass, M., Witkin, A., and Terzopoulos, D. Snakes: Active contour models. *Inter. J. Comput. Vision 1,* 4 (Jan. 1988), 321–331.
7. Klatt, D.H. Review of text-to-speech conversion for English. *J. Acoust. Soc. Am. 82,* 3 (1987), 737–793.
8. Klinker, G. VDI: A visual debugging interface for image interpretation. In *Visualization in Scientific Computing II,* F.H. Post, and A.J.S. Hin, Eds., Springer-Verlag, Berlin, 1992.
9. Lorensen, W. and Cline, H. Marching cubes: High resolution 3D surface construction algorithm. *Comput. Graph. 21,* 4 (1987), 163–169.

10. Pechura, C.M. and Martin, J.B. *Mapping the Brain and its Functions, Integrating Enabling Technologies in Neuroscience Research.* National Academy Press, Wash., D.C., 1991.
11. Sachs, E., Roberts, A., and Stoops, D. 3-Draw: A tool for designing 3D shapes. *IEEE Comput. Graph. Appli. 11,* 6 (Nov. 1991), 18–26.
12. Samet, H. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, Mass., 1989.
13. Sederberg, T.W., and Parry, S.R. Free-form deformations of solid geometric models. *Comput. Graph. (SIGGRAPH'86) 20,* 4 (Aug. 1986), 151–160.
14. Szeliski, R. Real-time octree generation from rotating objects. Tech. Rep. 90/12, Digital Equipment Corporation, Cambridge Research Lab, Dec. 1990.
15. Szeliski, R. Shape from rotation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (CVPR'91),* (Maui, Hawaii, June 1991), IEEE Computer Society Press, pp. 625–630.
16. Szeliski, R. and Tonnesen, D. Surface modeling with oriented particle systems. Tech. Rep. 91/14, Digital Equipment Corporation, Cambridge Research Lab, Dec. 1991. To be published in *SIGGRAPH '92 Proceedings.*
17. Upson, C., Faulhaber, T., Jr., Kamins, D., Laidlaw, D., Schlegel, D., Vroom, J., Gurwitz, R., and van Dam, A. The application visualization system: A computational environment for scientific visualization. *IEEE Comput. Graph. Appli. 9,* 4 (1989), 30–42.
18. Vannier, M., Pilgram, T., Bhatia, G., and Brunsden, B. Facial surface scanner. *IEEE Comput. Graph. Appli. 11,* 6 (1991), 72–80.
19. Waters, K. A muscle model for animating three-dimensional facial expressions. *Comput. Graph. (SIGGRAPH'87), 21,* 4 (July 1987), 17–24.
20. Waters, K. and Terzopoulos, D. A physical model of facial tissue and muscle articulation. In *Proceedings of the First Conference on Visualization in Biomedical Computing* (May 1990), pp.77–82.

CR Categories and Subject Descriptors: D.2.2 [**Software Engineering**]: Tools and Techniques—*User interfaces;* H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Audio input/output;* I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding—*Shape;* I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Physically based modeling;* I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Raytracing;* I.4.3 [**Image Processing**]: Enhancement—*registration;* I.4.6 [**Image Processing**]: Segmentation—*Edge and feature detection;* I.6.8 [**Simulation and Modeling**]: Types of simulation—*animation*

General Terms: Algorithms, Human Factors

Additional Key Words and Phrases: Facial animation, medical and biological imaging, octrees

About the Authors:

INGRID CARLBOM is manager of visualization research at the Cambridge Research Lab of Digital Equipment Corporation. Her research interests include scientific visualization, medical and biological imaging, geometric modeling, and computer graphics system architecture; email: carlbom@crl.dec.com

WILLIAM M HSU is a member of the resarch support staff at the Cambridge Research Lab of Digital Equipment Corporation. His research interests include scientific visualization, computer graphics, geometric modeling, and parallel algorithms; email: hsu@crl.dec.com

GUDRUN KLINKER is a member of the research staff at the Cambridge Research Lab of Digital Equipment Corporation. Her research interests include color computer vision and visualization environments for semiautomatic data interpretation. email: gudrun@crl.dec.com

RICHARD SZELISKI is a member of the research staff at the Cambridge Research Lab of Digital Equipment Corporation. His research interests include 3D computer vision, computer graphics, and parallel processing. email: szeliski@crl.dec.com

KEITH WATERS is a member of the research staff at the Cambridge Research Lab of Digital Equipment Corporation. His research interests include computer graphics, physically based modeling, volume visualization, medical facial applications, and facial synthesis. email: waters@crl.dec.com

Authors' Present Address:

Carlbom, Hsu, Klinker, Szeliski, Waters, Gettys, and Levergood: Digital Equipment Corporation, Cambridge Research Lab, 1 Kendall Square, Building 700, Cambridge, MA 02139; tel. (617) 621-6650; email: lastname@crl.dec.com

Doyle: Biomedical Visualization Lab (M/C 527), College of Associated Health Professionals, Univ. of Ill. at Chicago, 1919 West Taylor, RM 211, Chicago, ILL 60612; tel. (312) 996-7337; fax: 312-996-8342; email: u52838@uicvm.cc.uic.edu

Harris: Neurology Research Dept. Children's Hospital, 300 Longwood Ave., Boston MA 02115; tel. (617) 735-6373; fax: 617-730-0636

Palmer R. and Palmer L. and Wallace: Digital Equip. Corp. 146 Main St., Maynard, MA 01754-2571; tel. (617) 493-5111

Picart: Dept. of Electrical Engineering, Univ. of Mass. Lowell, 1 University Ave., Lowell, MA 01854

Terzopoulos and Tonnesen: Computer Science Dept., Univ. of Toronto, 10 Kings College Rd., Toronto, Ontario, Canada M5S 1A4; tel. (416) 978-7777; email: dt@ai.toronto.edu

Vannier: Mallinckrodt Institute of Radiology, Washington Univ. School of Medicine; 512 S. Kingshighway Blvd., St. Louis, MO 63110; tel. (314) 362-8467; fax 314 362-8491; email: mvannier@davey.wustl.edu

Acknowledgment

John C. Hart, guest editor of the special section on graphics extends his appreciation to the following: Tom DeFanti and Maxine Brown of the Electronic Visualization Laboratory (EVL) at the University of Chicago, and Jim George of Mesa Graphics, Inc. in, Los Alamos, N. Mex., were very helpful in putting this issue together. Thanks also to Charlie Catlett of the National Center for Supercomputing Applications, University of Illinois, Champaign Urbana, for information on Showcase's predecessors at Supercomputing conferences. Many of the color images were submitted electronically and reproduced on film in Chicago thanks to Gary Lindahl of the EVL and Mary Rasmussen of the Biomedical Visualization Laboratory at the University of Chicago. Kathy O'Keefe of the EVL produced the images of the virtual "Showcase" exhibition.

Parallel Database Systems: The Future of High Performance Database Systems

David DeWitt and Jim Gray

Highly parallel database systems are beginning to displace traditional mainframe computers for the largest database and transaction processing tasks. The success of these systems refutes a 1983 paper predicting the demise of database machines [3]. Ten years ago the future of highly parallel database machines seemed gloomy, even to their staunchest advocates. Most database machine research had focused on specialized, often trendy, hardware such as CCD memories, bubble memories, head-per-track disks, and optical disks. None of these technologies fulfilled their promises; so there was a sense that conventional CPUs, electronic RAM, and moving-head magnetic disks would dominate the scene for many years to come. At that time, disk throughput was predicted to double while processor speeds were predicted to increase by much larger factors. Consequently, critics predicted that multiprocessor systems would soon be I/O limited unless a solution to the I/O bottleneck was found.

While these predictions were fairly accurate about the future of hardware, the critics were certainly wrong about the overall future of parallel database systems. Over the last decade Teradata, Tandem, and a host of startup companies have successfully developed and marketed highly parallel machines.

Parallel

Why have parallel database systems become more than a research curiosity? One explanation is the widespread adoption of the relational data model. In 1983 relational database systems were just appearing in the marketplace; today they dominate it. Relational queries are ideally suited to parallel execution; they consist of uniform operations applied to uniform streams of data. Each operator produces a new relation, so the operators can be composed into highly parallel dataflow graphs. By streaming the output of one operator into the input of another operator, the two operators can work in series giving *pipelined parallelism*. By partitioning the input data among multiple processors and memories, an operator can often be split into many independent operators each working on a part of the data. This partitioned data and execution gives *partitioned parallelism* (Figure 1).

The dataflow approach to database system design needs a message-based client-server operating system to interconnect the parallel processes executing the relational operators. This in turn requires a high-speed network to interconnect the parallel processors. Such facilities seemed exotic a decade ago, but now they are the mainstream of computer architecture. The client-server paradigm using high-speed LANs is the basis for most PC, workstation, and workgroup software. Those same client-server mechanisms are an excellent basis for distributed database technology.

Mainframe designers have found it difficult to build machines powerful enough to meet the CPU and I/O demands of relational databases serving large numbers of simultaneous users or searching terabyte databases. Meanwhile, multiprocessors based on fast and inexpensive microprocessors have become widely available from vendors including Encore, Intel, NCR, nCUBE, Sequent, Tandem, Teradata, and Thinking Machines. These machines provide more total power than their mainframe coun-

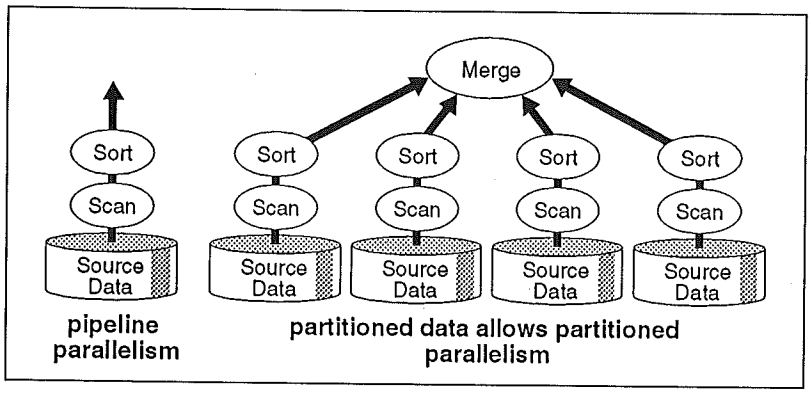


Figure 1.
The dataflow approach to relational operators gives both pipelined and partitioned parallelism. Relational data operators take relations (uniform sets of records) as input and produce relations as outputs. This allows them to be composed in dataflow graphs that allow *pipeline parallelism* (left) in which the computation of one operator proceeds in parallel with another, and *partitioned parallelism* in which operators (sort and scan in the diagram at the right) are replicated for each data source, and the replicas execute in parallel.

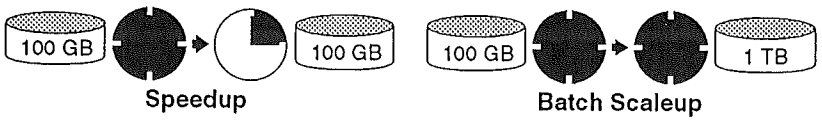


Figure 2.
Speedup and Scaleup. A speedup design performs a one-hour job four times faster when run on a four-times larger system. A scaleup design runs a ten-times bigger job is done in the same time by a ten-times bigger system.

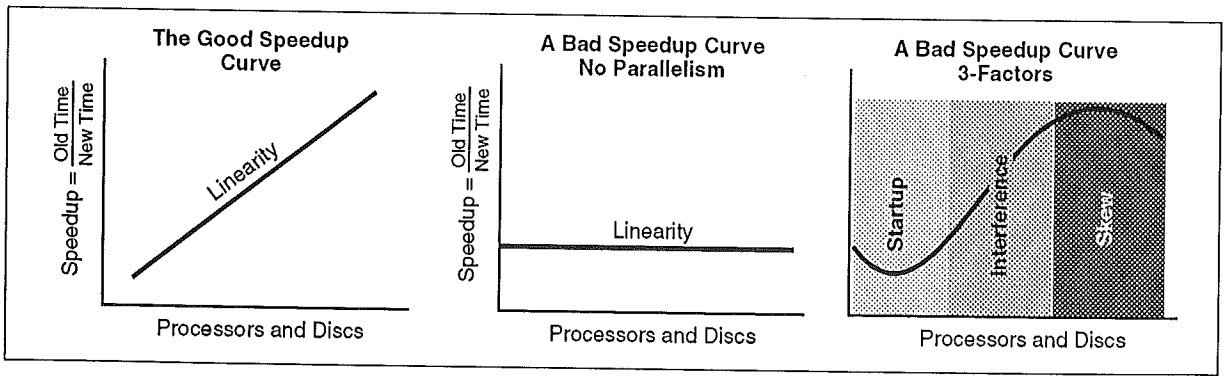


Figure 3.
Good and bad speedup curves. The standard speedup curves. The left curve is the ideal. The middle graph shows no speedup as hardware is added. The right curve shows the three threats to parallelism. Initial startup costs may dominate at first. As the number of processes increase, interference can increase. Ultimately, the job is divided so finely, that the variance in service times (skew) causes a slowdown.

terparts at a lower price. Their modular architectures enable systems to grow incrementally, adding MIPS, memory, and disks either to speedup the processing of a given job, or to scaleup the system to process a larger job in the same time.

In retrospect, special-purpose database machines have indeed failed; but parallel database systems are a big success. The successful parallel database systems are built from conventional processors, memories, and disks. They have emerged as major consumers of highly parallel architectures, and are in an excellent position to exploit massive numbers of fast-cheap commodity disks, processors, and memories promised by current technology forecasts.

A consensus on parallel and distributed database system architecture has emerged. This architecture is based on a *shared-nothing* hardware design [29] in which processors communicate with one another only by sending messages via an interconnection network. In such systems, tuples of each relation in the database are *partitioned (declustered)* across disk storage units[1] attached directly to each processor. Partitioning allows multiple processors to scan large relations in parallel without needing any exotic I/O devices. Such architectures were pioneered by Teradata in the late 1970s and by several research projects. This design is now used by Teradata, Tandem, NCR, OracleNCUBE, and several other products currently under development. The research community has also embraced this shared-nothing dataflow architecture in systems like Arbre, Bubba, and Gamma.

The remainder of this article is organized as follows: The next section describes the basic architectural concepts used in these parallel database systems. This is followed by a brief presentation of the unique features of the Teradata, Tandem, Bubba, and Gamma systems in the following section, entitled "The State of the Art." Several areas for future research are described in "Future Directions and Research Problems" prior to the conclusion of this article.

Basic Techniques for Parallel Database Machine Implementation

Parallelism Goals and Metrics: Speedup and Scaleup

The ideal parallel system demonstrates two key properties: (1) *linear speedup:* Twice as much hardware can perform the task in half the elapsed time, and (2) *linear scaleup:* Twice as much hardware can perform twice as large a task in the same elapsed time (see Figures 2 and 3).

More formally, given a fixed job run on a small system, and then run on a larger system, the *speedup* given by the larger system is measured as:

$$Speedup = \frac{small\_system\_elapsed\_time}{big\_system\_elapsed\_time}$$

Speedup is said to be linear, if an N-times large or more expensive system yields a speedup of N.

Speedup holds the problem size constant, and grows the system. Scaleup measures the ability to grow both the system and the problem. *Scaleup* is defined as the ability of an N-times larger system to perform an N-times larger job in the same elapsed time as the original system. The scaleup metric is:

$$Scaleup = \frac{small\_system\_elapsed\_time\_on\_small\_problem}{big\_system\_elapsed\_time\_on\_big\_problem}$$

If this scaleup equation evaluates to 1, then the scaleup is said to be linear[2]. There are two distinct kinds of scaleup, batch and transactional. If the job consists of performing many small independent requests submitted by many clients and operating on a shared database, then scaleup consists of N-times as many clients, submitting N-times as many requests against an N-times larger database. This is the scaleup typically found in transaction processing systems and timesharing systems. This form of scaleup is used by the Transaction Processing Performance Council to scaleup their transaction processing benchmarks [36]. Consequently, it is called *transaction scaleup.* Transaction scaleup is ideally suited to parallel systems since each transaction is typically a small independent job that can be run on a separate processor.

A second form of scaleup, called *batch scaleup,* arises when the scaleup task is presented as a single large job. This is typical of database queries and is also typical of scientific simulations. In these cases, scaleup consists of using an N-times larger computer to solve an N-times larger problem. For database systems batch scaleup translates to the same query on an N-times larger database; for scientific problems, batch scaleup translates to the same calculation on an N-times finer grid or on an N-times longer simulation.

The generic barriers to linear speedup and linear scaleup are the triple threats of:

startup: The time needed to start a parallel operation. If thousands of processes must be started, this can easily dominate the actual computation time.

interference: The slowdown each new process imposes on all others when accessing shared resources.

skew: As the number of parallel steps increases, the average size of each step decreases, but the variance can well exceed the mean. The service time of a job is the service time of the slowest step of the job. When the variance dominates the mean, increased parallelism improves elapsed time only slightly.

The subsection "A Parallel

[1]The term disk is used here as a shorthand for disk or other nonvolatile storage media. As the decade proceeds, nonvolatile electronic storage or some other media may replace or augment disks.

[2]The execution cost of some operators increases super-linearly. For example, the cost of sorting n-tuples increases as $nlog(n)$. When n is in the billions, scaling up by a factor of a thousand, causes $nlog(n)$ to increase by 3,000. This 30% deviation from linearity in a three-orders-of-magnitude scaleup justifies the use of the term *near-linear* scaleup.

Dataflow Approach to SQL Software" describes several basic techniques widely used in the design of shared-nothing parallel database machines to overcome these barriers. These techniques often achieve linear speedup and scaleup on relational operators.

Hardware Architecture, the Trend to Shared-Nothing Machines

The ideal database machine would have a single infinitely fast processor with an infinite memory with infinite bandwidth—and it would be infinitely cheap (free). Given such a machine, there would be no need for speedup, scaleup, or parallelism. Unfortunately, technology is not delivering such machines—but it is coming close. Technology is promising to deliver fast one-chip processors, fast high-capacity disks, and high-capacity electronic RAM. It also promises that each of these devices will be very inexpensive by today's standards, costing only hundreds of dollars each.

So, the challenge is to build an infinitely fast processor out of infinitely many processors of finite speed, and to build an infinitely large memory with infinite memory bandwidth from infinitely many storage units of finite speed. This sounds trivial mathematically; but in practice, when a new processor is added to most computer designs, it slows every other computer down just a little bit. If this slowdown (interference) is 1%, then the maximum speedup is 37 and a 1,000-processor system has 4% of the effective power of a single-processor system.

How can we build scaleable multiprocessor systems? Stonebraker suggested the following simple taxonomy for the spectrum of designs (see Figures 4 and 5) [29][3]:

shared-memory: All processors share direct access to a common global memory and to all disks. The IBM/370, Digital VAX, and Sequent Symmetry multiprocessors typify this design.

shared-disks: Each processor has a private memory but has direct access to all disks. The IBM Sysplex and original Digital VAXcluster typify this design.

shared-nothing: Each memory and disk is owned by some processor that acts as a server for that data. Mass storage in such an architecture is distributed among the processors by connecting one or more disks. The Teradata, Tandem, and nCUBE machines typify this design.

Shared-nothing architectures minimize interference by minimizing resource sharing. They also exploit commodity processors and memory without needing an incredibly powerful interconnection network. As Figure 5 suggests, the other architectures move large quantities of data through the interconnection network. The shared-nothing design moves only questions and answers through the network. Raw memory accesses and raw disk accesses are performed locally in a processor, and only the filtered (reduced) data is passed to the client program. This allows a more scaleable design by minimizing traffic on the interconnection network.

Shared-nothing characterizes the database systems being used by Teradata [33], Gamma [8, 9], Tandem [32], Bubba [1], Arbre [21], and nCUBE [13]. Significantly, Digital's VAXcluster has evolved to this design. DOS and UNIX workgroup systems from 3com, Borland, Digital, HP, Novell, Microsoft, and Sun also adopt a shared-nothing client-server architecture.

The actual interconnection networks used by these systems vary enormously. Teradata employs a redundant tree-structured communication network. Tandem uses a three-level duplexed network, two levels within a cluster, and rings connecting the clusters. Arbre, Bubba, and Gamma are independent of the underlying interconnection network, requiring only that the network allow any two nodes to communicate with one another. Gamma operates on an Intel Hypercube. The Arbre prototype was implemented using IBM 4381 processors connected to one another in a point-to-point network. Workgroup systems are currently making a transition from Ethernet to higher speed local networks.

The main advantage of shared-nothing multiprocessors is that they can be scaled up to hundreds and probably thousands of processors that do not interfere with one another. Teradata, Tandem, and Intel have each shipped systems with more than 200 processors. Intel is implementing a 2,000-node hypercube. The largest shared-memory multiprocessors currently available are limited to about 32 processors.

These shared-nothing architectures achieve near-linear speedups and scaleups on complex relational queries and on on-line transaction processing workloads [9, 10, 32]. Given such results, database machine designers see little justification for the hardware and software complexity associated with shared-memory and shared-disk designs.

Shared-memory and shared-disk systems do not scale well on database applications. Interference is a major problem for shared-memory multiprocessors. The interconnection network must have the bandwidth of the sum of the processors and disks. It is difficult to build such networks that can scale to thousands of nodes. To reduce network traffic and to minimize latency, each processor is given a large private cache. Measurements of shared-memory multiprocessors running database workloads show that loading and flushing these caches considerably degrades processor performance [35]. As parallelism increases, interference on shared resources limits performance. Multiprocessor systems

[3]Single Instruction stream, Multiple Data stream (SIMD) machines such as ILLIAC IV and its derivatives like MASSPAR and the "old" Connection Machine are ignored here because to date they have few successes in the database area. SIMD machines seem to have application in simulation, pattern matching, and mathematical search, but they do not seem to be appropriate for the multiuser, I/O intensive, and dataflow paradigm of database systems.

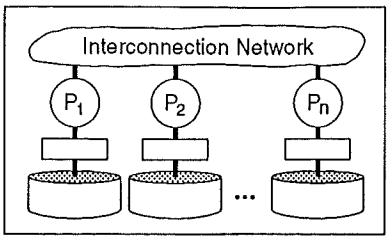


Figure 4.
The basic shared-nothing design. Each processor has a private memory and one or more disks. Processors communicate via a high-speed interconnect network. Teradata, Tandem, nCUBE, and the newer VAXclusters typify this design.

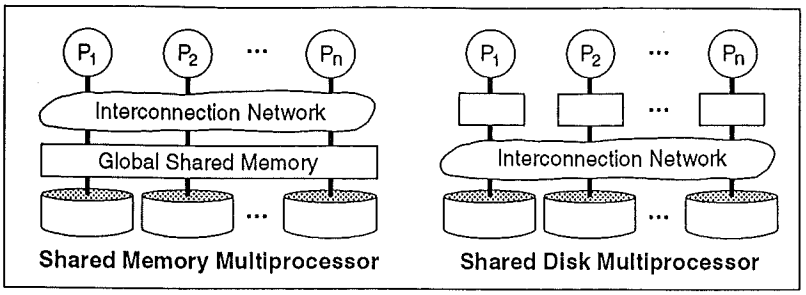


Figure 5.
The shared-memory and shared-disk designs. A shared-memory multiprocessor connects all processors to a globally shared memory. Multiprocessor IBM/370, VAX, and Sequent computers are typical examples of shared-memory designs. Shared-disk systems give each processor a private memory, but all the processors can directly address all the disks. Digital's VAXcluster and IBM's Sysplex typify this design.

Figure 6.
Example of a scan of a telephone relation to find the phone numbers of all people named Smith.

```
SELECT  telephone_number      /* the output attribute(s) */
FROM    telephone_book        /* the input relation      */
WHERE   last_name = 'Smith';  /* the predicate           */
```

often use an affinity scheduling mechanism to reduce this interference; giving each process an affinity to a particular processor. This is a form of data partitioning; it represents an evolutionary step toward the shared-nothing design. Partitioning a shared-memory system creates many of the skew and load balancing problems faced by a shared-nothing machine; but reaps none of the simpler hardware interconnect benefits. Based on this experience, we believe high-performance shared-memory machines will not economically scale beyond a few processors when running database applications.

To ameliorate the interference problem, most shared-memory multiprocessors have adopted a shared-disk architecture. This is the logical consequence of affinity scheduling. If the disk interconnection network can scale to thousands of disks and processors, then a shared-disk design is adequate for large read-only databases and for databases where there is no concurrent sharing. The shared-disk architecture is not very effective for database applications that read and write a shared database. A processor wanting to update some data must first obtain the current copy of that data. Since others might be updating the same data concurrently, the processor must declare its intention to update the data. Once this declaration has been honored and acknowledged by all the other processors, the updator can read the shared data from disk and update it. The processor must then write the shared data to disk so that subsequent readers and writers will be aware of the update. There are many optimizations of this protocol, but they all end up exchanging reservation messages and exchanging large physical data pages. This creates processor interference and delays. It creates heavy traffic on the shared interconnection network.

For shared database applications, the shared-disk approach is much more expensive than the shared-nothing approach of exchanging small high-level logical questions and answers among clients and servers. One solution to this interference has been to give data a processor affinity; other processors wanting to access the data send messages to the server managing the data. This has emerged as a major application of transaction processing monitors that partition the load among partitioned servers, and is also a major application for remote procedure calls. Again, this trend toward the partitioned data model and shared-nothing architecture on a shared-disk system reduces interference. Since the shared-disk system interconnection network is difficult to scale to thousands of processors and disks, many conclude that it would be better to adopt the shared-nothing architecture from the start.

Given the shortcomings of shared-disk and shared-memory architectures, why have computer architects been slow to adopt the shared-nothing approach? The first answer is simple, high-performance, low-cost commodity components have only recently become available. Traditionally, commodity components provided relatively low performance and low quality.

Today, old software is the most significant barrier to the use of parallelism. Old software written for uniprocessors gets no speedup or scaleup when put on any kind of multiprocessor. It must be rewritten to benefit from parallel processing and multiple disks. Database applications are a unique exception to this. Today, most database programs are written in the relational language SQL that has been standardized by both ANSI and ISO. It is possible to take standard SQL applications written for uniprocessor systems and execute them in parallel on shared-nothing database

machines. Database systems can automatically distribute data among multiple processors. Teradata and Tandem routinely port SQL applications to their system and demonstrate near-linear speedups and scaleups. The following subsection explains the basic techniques used by such parallel database systems.

A Parallel Dataflow Approach to SQL Software

Terabyte on-line databases, consisting of billions of records, are becoming common as the price of on-line storage decreases. These databases are often represented and manipulated using the SQL relational model. The next few paragraphs give a rudimentary introduction to relational model concepts needed to understand the remainder of this article.

A relational database consists of *relations* (*files* in COBOL terminology) that in turn contain *tuples* (*records* in COBOL terminology). All the tuples in a relation have the same set of *attributes* (*fields* in COBOL terminology).

Relations are created, updated, and queried by writing SQL statements. These statements are syntactic sugar for a simple set of operators chosen from the relational algebra. *Select-project,* here called *scan,* is the simplest and most common operator—it produces a row-and-column subset of a relational table. A scan of relation R using predicate P and attribute list L produces a relational data stream as output. The scan reads each tuple, t, of R and applies the predicate P to it. If $P(t)$ is true, the scan discards any attributes of t not in L and inserts the resulting tuple in the scan output stream. Expressed in SQL, a scan of a telephone book relation to find the phone numbers of all people named Smith would be written as shown in Figure 6. A scan's output stream can be sent to another relational operator, returned to an application, displayed on a terminal, or printed in a report. Therein lies the beauty and utility of the relational model. The uniformity of the data and operators allow them to be arbitrarily composed into dataflow graphs.

The output of a scan may be sent to a *sort* operator that will reorder the tuples based on an attribute sort criteria, optionally eliminating duplicates. SQL defines several *aggregate* operators to summarize attributes into a single value, for example, taking the sum, min, or max of an attribute, or counting the number of distinct values of the attribute. The *insert* operator adds tuples from a stream to an existing relation. The *update* and *delete* operators alter and delete tuples in a relation matching a scan stream.

The relational model defines several operators to combine and compare two or more relations. It provides the usual set operators *union, intersection, difference,* and some more exotic ones like *join* and *division.* Discussion here will focus on the *equi-join* operator (here called *join*). The join operator composes two relations, A and B, on some attribute to produce a third relation. For each tuple, *ta*, in A, the join finds all tuples, *tb*, in B whose attribute values are equal to that of *ta*. For each matching pair of tuples, the join operator inserts into the output stream a tuple built by concatenating the pair.

Codd, in a classic paper, showed that the relational data model can represent any form of data, and that these operators are complete [5]. Today, SQL applications are typically a combination of conventional programs and SQL statements. The programs interact with clients, perform data display, and provide high-level direction of the SQL dataflow.

The SQL data model was originally proposed to improve programmer productivity by offering a nonprocedural database language. Data independence was an additional benefit; since the programs do not specify how the query is to be executed, SQL programs continue to operate as the logical and physical database schema evolves.

Parallelism is an unanticipated benefit of the relational model. Since relational queries are really just relational operators applied to very large collections of data, they offer many opportunities for parallelism. Since the queries are presented in a nonprocedural language, they offer considerable latitude in executing the queries.

Relational queries can be executed as a dataflow graph. As mentioned in the first section of this article, these graphs can use both pipelined parallelism and partitioned parallelism. If one operator sends its output to another, the two operators can execute in parallel giving potential speedup of two.

The benefits of pipeline parallelism are limited because of three factors: (1) Relational pipelines are rarely very long—a chain of length ten is unusual. (2) Some relational operators do not emit their first output until they have consumed all their inputs. Aggregate and sort operators have this property. One cannot pipeline these operators. (3) Often, the execution cost of one operator is much greater than the others (this is an example of skew). In such cases, the speedup obtained by pipelining will be very limited.

Partitioned execution offers much better opportunities for speedup and scaleup. By taking the large relational operators and partitioning their inputs and outputs, it is possible to use divide-and-conquer to turn one big job into many independent little ones. This is an ideal situation for speedup and scaleup. Partitioned data is the key to partitioned execution.

Data Partitioning. Partitioning a relation involves distributing its tuples over several disks. Data partitioning has its origins in centralized systems that had to partition files, either because the file was too big for one disk, or because the file access rate could not be supported by a single disk. Distributed databases use data partitioning when they place relation fragments at different network sites [23]. Data par-

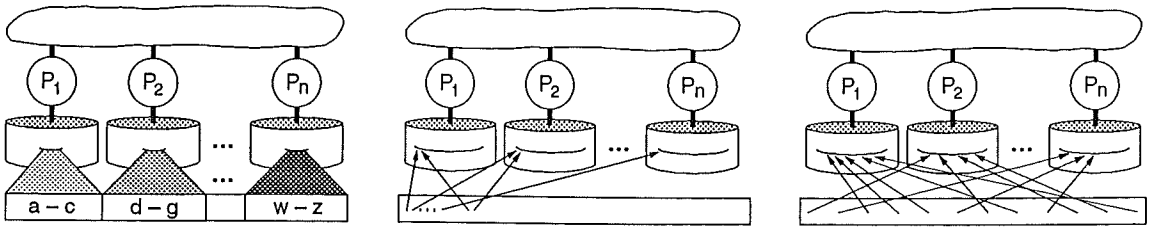


Figure 7.
The three basic partitioning schemes. Range partitioning maps contiguous attribute ranges of a relation to various disks. Round-robin partitioning maps the i'th tuple to disk i **mod** n. Hashed partitioning maps each tuple to a disk location based on a hash function. Each of these schemes spreads data among a collection of disks, allowing parallel disk access and parallel processing.

titioning allows parallel database systems to exploit the I/O bandwidth of multiple disks by reading and writing them in parallel. This approach provides I/O bandwidth superior to RAID-style systems without needing any specialized hardware [22, 24].

The simplest partitioning strategy distributes tuples among the fragments in a *round-robin* fashion. This is the partitioned version of the classic entry-sequence file. Round-robin partitioning is excellent if all applications want to access the relation by sequentially scanning all of it on each query. The problem with round-robin partitioning is that applications frequently want to associatively access tuples, meaning that the application wants to find all the tuples having a particular attribute value. The SQL query looking for the Smiths in the phone book shown in Figure 6 is an example of an associative search.

Hash partitioning is ideally suited for applications that want only sequential and associative access to the data. Tuples are placed by applying a *hashing* function to an attribute of each tuple. The function specifies the placement of the tuple on a particular disk. Associative access to the tuples with a specific attribute value can be directed to a single disk, avoiding the overhead of starting queries on multiple disks. Hash partitioning mechanisms are provided by Arbre, Bubba, Gamma, and Teradata.

Database systems pay considerable attention to clustering related data together in physical storage. If a set of tuples is routinely accessed together, the database system attempts to store them on the same physical page. For example, if the Smiths of the phone book are routinely accessed in alphabetical order, then they should be stored on pages in that order, these pages should be clustered together on disk to allow sequential prefetching and other optimizations. Clustering is very application-specific. For example, tuples describing nearby streets should be clustered together in geographic databases, tuples describing the line items of an invoice should be clustered with the invoice tuple in an inventory control application.

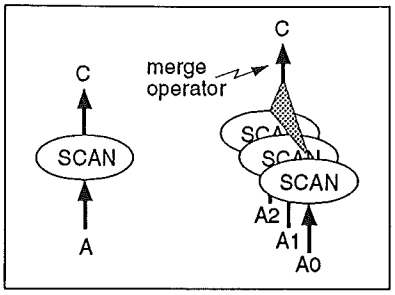


Figure 8.
Partitioned data parallelism. A simple relational dataflow graph showing a relational scan (project and select) decomposed into three scans on three partitions of the input stream or relation. These three scans send their output to a merge node that produces a single data stream.

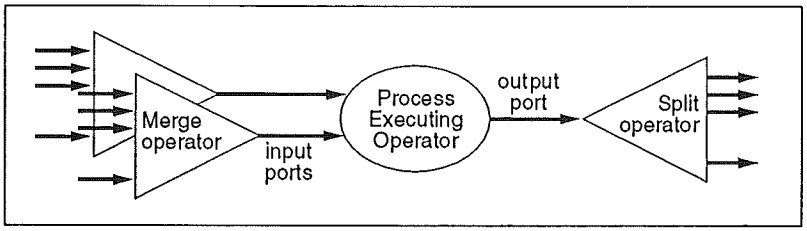


Figure 9.
Merging the inputs and partitioning the output of an operator. A relational dataflow graph showing a relational operator's inputs being merged to a sequential stream per port. The operator's output is being decomposed by a split operator in several independent streams. Each stream may be a duplicate or a partitioning of the operator output stream into many disjoint streams. With the split and merge operators, a web of simple sequential dataflow nodes can be connected to form a parallel execution plan.

Hashing tends to randomize data rather than cluster it. *Range partitioning* clusters tuples with similar attributes together in the same partition. It is good for sequential and associative access, and is also good for clustering data. Figure 7 shows range partitioning based on lexicographic order, but any clustering algorithm is possible. Range partitioning derives its name from the typical SQL range queries such as

```
latitude BETWEEN 38° AND 39°
```

Arbre, Bubba, Gamma, Oracle, and Tandem provide range partitioning.

The problem with range partitioning is that it risks *data skew*, where all the data is placed in one partition, and *execution skew* in which all the execution occurs in one partition. Hashing and round-robin are less susceptible to these skew problems. Range partitioning can minimize skew by picking non-uniformly-distributed partitioning criteria. Bubba uses this concept by considering the access frequency *(heat)* of each tuple when creating partitions of a relation; the goal being to balance the frequency with which each partition is accessed (its *temperature*) rather than the actual number of tuples on each disk (its volume) [6].

Table 1.
Sample Split Operators.
Each split operator maps tuples to a set of output streams (ports of other processes) depending on the range value (predicate) of the input tuple. The split operator on the left is for the relation A scan in Figure 10, while the table on the right is for the relation B scan. The tables partition the tuples among three data streams.

Relation A Scan Split Operator		Relation B Scan Split Operator	
Predicate	Destination Process	Predicate	Destination Process
"A-H"	(CPU #5, Process #3, Port #0)	"A-H"	(CPU #5, Process #3, Port #1)
"I-Q"	(CPU #7, Process #8, Port #0)	"I-Q"	(CPU #7, Process #8, Port #1)
"R-Z"	(CPU #2, Process #2, Port #0)	"R-Z"	(CPU #2, Process #2, Port #1)

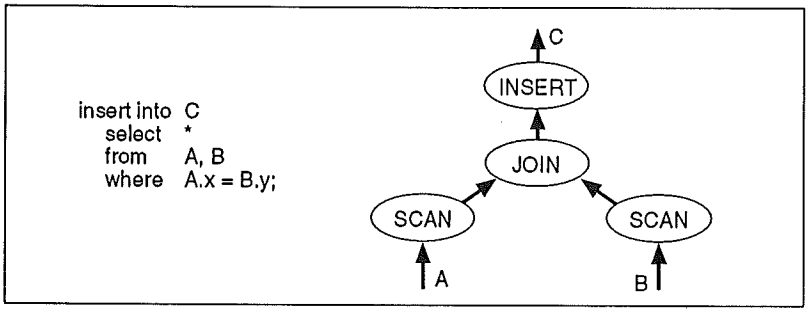


Figure 10.
A simple SQL query and the associated relational query graph. The query specifies that a join is to be performed between relations A and B by comparing the x attribute of each tuple from the A relation with the y attribute value of each tuple of the B relation. For each pair of tuples that satisfy the predicate, a result tuple is formed from all the attributes of both tuples. This result tuple is then added to the result relation C. The associated logical query graph (as might be produced by a query optimizer) shows a tree of operators, one for the join, one for the insert, and one for scanning each input relation.

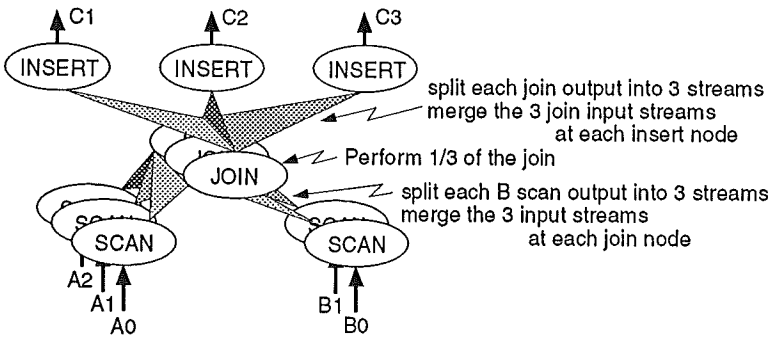


Figure 11.
A simple relational dataflow graph. It shows two relational scans (project and select) consuming two input relations, A and B and feeding their outputs to a join operator that in turn produces a data stream C.

While partitioning is a simple concept that is easy to implement, it raises several new physical database design issues. Each relation must now have a partitioning strategy and a set of disk fragments. Increasing the degree of partitioning usually reduces the response time for an individual query and increases the overall throughput of the system. For sequential scans, the response time decreases because more processors and disks are used to execute the query. For associative scans, the response time improves because fewer tuples are stored at each node and hence the size of the index that must be searched decreases.

There is a point beyond which further partitioning actually increases the response time of a query. This point occurs when the cost of starting a query on a node becomes a significant fraction of the actual execution time [6, 11].

Parallelism Within Relational Operators. Data partitioning is the first step in partitioned execution of relational dataflow graphs. The basic idea is to use parallel data streams instead of writing new parallel operators (programs). This approach enables the use of unmodified, existing sequential routines to execute the relational operators in parallel. Each relational operator has a set of *input ports* on which input tuples arrive and an *output* port to which the operator's output stream is sent. The parallel dataflow works by partitioning and merging data streams into these sequential ports. This approach allows the use of existing sequential relational operators to execute in parallel.

Consider a scan of a relation, A, that has been partitioned across three disks into fragments A0, A1, and A2. This scan can be imple-

mented as three scan operators that send their output to a common merge operator. The merge operator produces a single output data stream to the application or to the next relational operator. The parallel query executor creates the three scan processes shown in Figure 8 and directs them to take their inputs from three different sequential input streams (A0, A1, A2). It also directs them to send their outputs to a common merge node. Each scan can run on an independent processor and disk. So the first basic parallelizing operator is a *merge* that can combine several parallel data streams into a single sequential stream.

The merge operator tends to focus data on one spot. If a multistage parallel operation is to be done in parallel, a single data stream must be split into several independent streams. A *split operator* is used to partition or replicate the stream of tuples produced by a relational operator. A split operator defines a mapping from one or more attribute values of the output tuples to a set of destination processes (see Figure 9).

As an example, consider the two split operators shown in Table 1 in conjunction with the SQL query shown in Figure 10. Assume that three processes are used to execute the join operator, and that five other processes execute the two scan operators—three scanning partitions of relation A while two scan partitions of relation B. Each of the three relation A scan nodes will have the same split operator, sending all tuples between "A-H" to port 1 of join process 0, all between "I-Q" to port 1 of join process 1, and all between "R-Z" to port 1 of join process 2. Similarly the two relation B scan nodes have the same split operator except that their outputs are merged by port 1 (not port 0) of each join process. Each join process sees a sequential input stream of A tuples from the port 0 merge (the left-scan nodes) and another sequential stream of B tuples from the port 1 merge (the right-scan nodes). The outputs of each join are, in turn, split into three streams based on the partitioning criterion of relation C.

To clarify this example, consider the first join process in Figure 11 (processor 5, process 3, ports 0 and 1 in Table 1). It will receive all the relation A "A-H" tuples from the three relation A scan operators merged as a single stream on port 0, and will get all the "A-H" tuples from relation B merged as a single stream on port 1. It will join them using a hash-join, sort-merge join, or even a nested join if the tuples arrive in the proper order.

If each of these processes is on an independent processor with an independent disk, there will be little interference among them. Such dataflow designs are a natural application for shared-nothing machine architectures.

The split operator in Table 1 is just an example. Other split operators might duplicate the input stream, or partition it round-robin, or partition it by hash. The partitioning function can be an arbitrary program. Gamma, Volcano, and Tandem use this approach [14]. It has several advantages including the automatic parallelism of any new operator added to the system, plus support for many kinds of parallelism.

The split and merge operators have flow control and buffering built into them. This prevents one operator from getting too far ahead in the computation. When a split-operator's output buffers fill, it stalls the relational operator until the data target requests more output.

For simplicity, these examples have been stated in terms of an operator per process. But it is entirely possible to place several operators within a process to get coarser grained parallelism. The fundamental idea though is to build a self-pacing dataflow graph and distribute it in a shared-nothing machine in a way that minimizes interference.

Specialized Parallel Relational Operators. Some algorithms for relational operators are especially appropriate for parallel execution, either because they minimize data flow, or because they better tolerate data and execution skew. Improved algorithms have been found for most of the relational operators. The evolution of join operator algorithms is sketched here as an example of these improved algorithms.

Recall that the join operator combines two relations, A and B, to produce a third relation containing all tuple pairs from A and B with matching attribute values. The conventional way of computing the join is to sort both A and B into new relations ordered by the join attribute. These two intermediate relations are then compared in sorted order, and matching tuples are inserted in the output stream. This algorithm is called *sort-merge* join.

Many optimizations of sort-merge join are possible, but since sort has execution cost *nlog(n)*, sort-merge join has an *nlog(n)* execution cost. Sort-merge join works well in a parallel dataflow environment unless there is data skew. In case of data skew, some sort partitions may be much larger than others. This in turn creates execution skew and limits speedup and scaleup. These skew problems do not appear in centralized sort-merge joins.

Hash-join is an alternative to sort-merge join. It has linear execution cost rather than *nlog(n)* execution cost, and it is more resistant to data skew. It is superior to sort-merge join unless the input streams are already in sorted order. Hash join works as follows. Each of the relations A and B are first hash partitioned on the join attribute. A hash partition of relation A is hashed into memory. The corresponding partition of table relation B is scanned, and each tuple is compared against the main-memory hash table for the A partition. If there is a match, the pair of tuples are sent to the output stream. Each pair of hash partitions is compared

in this way.

The hash join algorithm breaks a big join into many little joins. If the hash function is good and if the data skew is not too bad, then there will be little variance in the hash bucket size. In these cases hash-join is a linear-time join algorithm with linear speedup and scaleup. Many optimizations of the parallel hash-join algorithm have been discovered over the last decade. In pathological skew cases, when many or all tuples have the same attribute value, one bucket may contain all the tuples. In these cases no algorithm is known to speedup or scaleup.

The hash-join example shows that new parallel algorithms can improve the performance of relational operators. This is a fruitful research area [4, 8, 18, 20, 25, 26, 38, 39]. Although parallelism can be obtained from conventional sequential relational algorithms by using split and merge operators, we expect that many new algorithms will be discovered in the future.

The State of the Art

Teradata

Teradata quietly pioneered many of the ideas presented in this article. Since 1978 they have been building shared-nothing highly-parallel SQL systems based on commodity microprocessors, disks, and memories. Teradata systems act as SQL servers to client programs operating on conventional computers.

Teradata systems may have over 1,000 processors and many thousands of disks. The Teradata processors are functionally divided into two groups: Interface Processors (IFPs) and Access Module Processors (AMPs). The IFPs handle communication with the host, query parsing and optimization, and coordination of AMPs during query execution. The AMPs are responsible for executing queries. Each AMP typically has several disks and a large memory cache. IFPs and AMPs are interconnected by a dual redundant, tree-shaped interconnect called the Y-net [33].

Each relation is hash partitioned over a subset of the AMPs. When a tuple is inserted into a relation, a hash function is applied to the primary key of the tuple to select an AMP for storage. Once a tuple arrives at an AMP, a second hash function determines the tuple's placement in its fragment of the relation. The tuples in each fragment are in hash-key order. Given a value for the key attribute, it is possible to locate the tuple in a single AMP. The AMP examines its cache, and if the tuple is not present, fetches it in a single disk read. Hash secondary indices are also supported.

Hashing is used to split the outputs of relational operators into intermediate relations. Join operators are executed using a parallel sort-merge algorithm. Rather than using pipelined parallel execution, during the execution of a query, each operator is run to completion on all participating nodes before the next operator is initiated.

Teradata has installed many systems containing over 100 processors and hundreds of disks. These systems demonstrate near-linear speedup and scaleup on relational queries, and far exceed the speed of traditional mainframes in their ability to process large (terabyte) databases.

Tandem NonStop SQL

The Tandem NonStop SQL system is composed of processor clusters interconnected via 4-plexed fiber-optic rings. Unlike most other systems discussed in this article, the Tandem systems run the applications on the same processors and operating system as the database servers. There is no front-end/back-end distinction between programs and machines. The systems are configured at a disk per MIPS, so each 10-MIPS processor has about 10 disks. Disks are typically duplexed [2]. Each disk is served by a set of processes managing a large shared RAM cache, a set of locks, and log records for the data on that disk pair. Considerable effort is spent on optimizing sequential scans by prefetching large units, and by filtering and manipulating the tuples with SQL predicates at these disk servers. This minimizes traffic on the shared interconnection network.

Relations may be range partitioned across multiple disks. Entry-sequenced, relative, and B-tree organizations are supported. Only B-tree secondary indices are supported. Nested join, sort-merge join, and hash join algorithms are provided. Parallelization of operators in a query plan is achieved by inserting split and merge operators between operator nodes in the query tree. Scans, aggregates, joins, updates, and deletes are executed in parallel. In addition, several utilities use parallelism (e.g., load, reorganize, . . .) [31, 39].

Tandem systems are primary designed for on-line transaction processing (OLTP)—running many simple transactions against a large shared database. Beyond the parallelism inherent in running many independent transactions in parallel, the main parallelism feature for OLTP is parallel index update. SQL relations typically have five indices on them, although it is not uncommon to see 10 indices on a relation. These indices speed reads, but slow down inserts, updates, and deletes. By doing the index maintenance in parallel, the maintenance time for multiple indices can be held almost constant if the indices are spread among many processors and disks.

Overall, the Tandem systems demonstrate near-linear scaleup on transaction processing workloads, and near-linear speedup and scaleup on large relational queries [10, 31].

Gamma

The current version of Gamma runs on a 32-node Intel iPSC/2 Hypercube with a disk attached to each node. In addition to round-robin, range and hash partitioning, Gamma also provides hybrid-range

partitioning that combines the best features of the hash and range partitioning strategies [12]. Once a relation has been partitioned, Gamma provides both clustered and nonclustered indices on either the partitioning or nonpartitioning attributes. The indices are implemented as B-trees or hash tables.

Gamma uses split and merge operators to execute relational algebra operators using both parallelism and pipelining [9]. Sort-merge and three different hash join methods are supported [7]. Near-linear speedup and scaleup for relational queries has been measured on this architecture [9, 25, 26].

The Super Database Computer

The Super Database Computer (SDC) project at the University of Tokyo presents an interesting contrast to other database systems [16, 20]. SDC takes a combined hardware and software approach to the performance problem. The basic unit, called a processing module (PM), consists of one or more processors on a shared memory. These processors are augmented by a special-purpose sorting engine that sorts at high speed (3MB/second at present), and by a disk subsystem [19]. Clusters of processing modules are connected via an omega network that provides both nonblocking NxN interconnect and some dynamic routing minimize skewed data distribution during hash joins. The SDC is designed to scale to thousands of PMs, and so considerable attention is paid to the problem of data skew.

Data is partitioned among the PMs by hashing. The SDC software includes a unique operating system, and a relational database query executor. The SDC is a shared-nothing design with a software dataflow architecture. This is consistent with our assertion that current parallel database machines systems use conventional hardware. But the special-purpose design of the omega network and of the hardware sorter clearly contradict the thesis that special-purpose hardware is not a good investment of development resources. Time will tell whether these special-purpose components offer better price performance or peak performance than shared-nothing designs built of conventional hardware.

Bubba

The Bubba prototype was implemented using a 40-node FLEX/32 multiprocessor with 40 disks [4]. Although this is a shared-memory multiprocessor, Bubba was designed as a shared-nothing system and the shared-memory is only used for message passing. Nodes are divided into three groups: Interface Processors for communicating with external host processors and coordinating query execution; Intelligent Repositories for data storage and query execution; and Checkpoint/Logging Repositories. While Bubba also uses partitioning as a storage mechanism (both range and hash partitioning mechanisms are provided) and dataflow processing mechanisms, Bubba is unique in several ways. First, Bubba uses FAD rather than SQL as its interface language. FAD is an extended-relational persistent programming language. FAD provides support for complex objects via several type constructors including shared subobjects, set-oriented data manipulation primitives, and more traditional language constructs. The FAD compiler is responsible for detecting operations that can be executed in parallel according to how the data objects being accessed are partitioned. Program execution is performed using a dataflow execution paradigm. The task of compiling and parallelizing a FAD program is significantly more difficult than parallelizing a relational query. Another Bubba feature is its use of a single-level store mechanism in which the persistent database at each node is mapped to the virtual memory address space of each process executing at the node. This is in contrast to the traditional approach of files and pages. Similar mechanisms are used in IBM's AS400 mapping of SQL databases into virtual memory, HP's mapping of the Image Database into the operating system virtual address space, and Mach's mapped file [34] mechanism. This approach simplified the implementation of the upper levels of the Bubba software.

Other Systems

Other parallel database system prototypes include XPRS [30], Volcano [14], Arbre [21], and the PERSIST project under development at IBM Research Labs in Hawthorne and Almaden. While both Volcano and XPRS are implemented on shared-memory multiprocessors, XPRS is unique in its exploitation of the availability of massive shared-memory in its design. In addition, XPRS is based on several innovative techniques for obtaining extremely high performance and availability.

Recently, the Oracle database system has been implemented atop a 64-node nCUBE shared-nothing system. The resulting system is the first to demonstrate more than 1,000 transactions per second on the industry-standard TPC-B benchmark. This is far in excess of Oracle's performance on conventional mainframe systems—both in peak performance and in price/performance [13].

The NCR Corporation has announced the 3600 and 3700 product lines that employ shared-nothing architectures running System V R4 of Unix on Intel 486 and 586 processors. The interconnection network for the 3600 product line uses an enhanced Y-Net licensed from Teradata while the 3700 is based on a new multistage interconnection network being developed jointly by NCR and Teradata. Two software offerings have been announced. The first, a port of the Teradata software to a Unix environment, is targeted toward the decision-support marketplace. The second, based on a parallelization of the Sybase DBMS, is intended primarily for transaction processing workloads.

Database Machines and Grosch's Law

Today shared-nothing database machines have the best peak performance and best price performance available. When compared to traditional mainframes, the Tandem system scales linearly well beyond the largest reported mainframes on the TPC-A transaction processing benchmark. Its price/performance on these benchmarks is three times cheaper than the comparable mainframe numbers. Oracle on an nCUBE has the highest reported TPC-B numbers, and has very competitive price performance [13, 36]. These benchmarks demonstrate linear scaleup on transaction processing benchmarks.

Gamma, Tandem, and Teradata have demonstrated linear speedup and scaleup on complex relational database benchmarks. They scale well beyond the size of the largest mainframes. Their performance and price performance is generally superior to mainframe systems.

These observations defy Grosch's law. In the 1960s, Herb Grosch observed that there is an economy-of-scale in computing. At that time, expensive computers were much more powerful than inexpensive computers. This gave rise to super-linear speedups and scaleups. The current pricing of mainframes at $25,000/MIPS and $1,000/MB of RAM reflects this view. Meanwhile, microprocessors are selling for $250/MIPS and $100/MB of RAM.

By combining hundreds or thousands of these small systems, one can build an incredibly powerful database machine for much less money than the cost of a modest mainframe. For database problems, the near-linear speedup and scaleup of these shared-nothing machines allows them to outperform current shared-memory and shared disk mainframes.

Grosch's law no longer applies to database and transaction processing problems. There is no economy of scale. At best, one can expect linear speedup and scaleup of performance and price/performance. Fortunately, shared-nothing database architectures achieve this near-linear performance.

Future Directions and Research Problems

Mixing Batch and OLTP Queries

The second section of this article, "Basic Techniques for Parallel Database Machine Implementation", concentrated on the basic techniques used for processing complex relational queries in a parallel database system. Concurrently running a mix of both simple and complex queries concurrently presents several unsolved problems.

One problem is that large relational queries tend to acquire many locks and tend to hold them for a relatively long time. This prevents concurrent updates of the data by simple on-line transactions. Two solutions are currently offered: give the ad-hoc queries a fuzzy picture of the database, not locking any data as they browse it. Such a "dirty-read" solution is not acceptable for some applications. Several systems offer a versioning mechanism that gives readers a consistent (old) version of the database while updators are allowed to create newer versions of objects. Other, perhaps better, solutions for this problem may also exist.

Priority scheduling is another mixed-workload problem. Batch jobs have a tendency to monopolize the processor, flood the memory cache, and make large demands on the I/O subsystem. It is up to the underlying operating system to quantize and limit the resources used by such batch jobs to ensure short response times and low variance in response times for short transactions. A particularly difficult problem is the *priority inversion problem,* in which a low-priority client makes a request to a high-priority server. The server must run at high priority because it is managing critical resources. Given this, the work of the low-priority client is effectively promoted to high priority when the low-priority request is serviced by the high-priority server. There have been several ad-hoc attempts at solving this problem, but considerably more work is needed.

Parallel Query Optimization

Current database query optimizers do not consider all possible plans when optimizing a relational query. While cost models for relational queries running on a single processor are now well-understood [27] they still depend on cost estimators that are a guess at best. Some dynamically select from among several plans at run time depending on, for example, the amount of physical memory actually available and the cardinalities of the intermediate results [15]. To date, no query optimizers consider all the parallel algorithms for each operator and all the query tree organizations. More work is needed in this area.

Another optimization problem relates to highly skewed value distributions. Data skew can lead to high variance in the size of intermediate relations, leading to both poor query plan cost estimates and sub-linear speedup. Solutions to this problem are an area of active research [17, 20, 37, 38].

Application Program Parallelism

The parallel database systems offer parallelism within the database system. Missing are tools to structure application programs to take advantage of parallelism inherent in these parallel systems. While automatic parallelization of applications programs written in COBOL may not be feasible, library packages to facilitate explicitly parallel application programs are needed. Ideally the SPLIT and MERGE operators could be packaged so that applications could benefit from them.

Physical Database Design

For a given database and workload there are many possible indexing and partitioning combinations. Database design tools are needed to help the database administrator select among these many design options. Such tools might accept as input a description of the queries comprising the workload, their frequency of execution, statistical information about the relations in the database, and a description of the

processors and disks. The resulting output would suggest a partitioning strategy for each relation plus the indices to be created on each relation. Steps in this direction are beginning to appear.

Current algorithms partition relations using the values of a single attribute. For example, geographic records could be partitioned by longitude or latitude. Partitioning on longitude allows selections for a longitude range to be localized to a limited number of nodes, selections on latitude must be sent to all the nodes. While this is acceptable in a small configuration, it is not acceptable in a system with thousands of processors. Additional research is needed on multidimensional partitioning and search algorithms.

On-line Data Reorganization and Utilities

Loading, reorganizing, or dumping a terabyte database at a megabyte per second takes over 12 days and nights. Clearly parallelism is needed if utilities are to complete within a few hours or days. Even then, it will be essential that the data be available while the utilities are operating. In the SQL world, typical utilities create indices, add or drop attributes, add constraints, and physically reorganize the data, changing its clustering.

One unexplored and difficult problem is how to process database utility commands while the system remains operational and the data remains available for concurrent reads and writes by others. The fundamental properties of such algorithms are that they must be *on-line* (operate without making data unavailable), *incremental* (operate on parts of a large database), *parallel* (exploit parallel processors), and *recoverable* (allow the operation to be canceled and return to the old state).

Summary and Conclusions

Like most applications, database systems want cheap, fast hardware. Today that means commodity processors, memories, and disks. Consequently, the hardware concept of a *database machine* built of exotic hardware is inappropriate for current technology. On the other hand, the availability of fast microprocessors, and small inexpensive disks packaged as standard inexpensive but fast computers is an ideal platform for *parallel database systems.* A shared-nothing architecture is relatively straightforward to implement and, more importantly, has demonstrated both speedup and scaleup to hundreds of processors. Furthermore, shared-nothing architectures actually simplify the software implementation. If the software techniques of data partitioning, dataflow, and intra-operator parallelism are employed, the task of converting an existing database management system to a highly parallel one becomes relatively straightforward. Finally, there are certain applications (e.g., data mining in terabyte databases) that require the computational and I/O resources available only from a parallel architecture.

While the successes of both commercial products and prototypes demonstrate the viability of highly parallel database machines, several research issues remain unsolved including techniques for mixing ad-hoc queries with on-line transaction processing without seriously limiting transaction throughput, improved optimizers for parallel queries, tools for physical database design, on-line database reorganization, and algorithms for handling relations with highly skewed data distributions. Some application domains are not well supported by the relational data model. It appears that a new class of database systems based on an object-oriented data model is needed. Such systems pose a host of interesting research problems that require further examination. ◘

References

1. Alexander, W., et al. Process and dataflow control in distributed data-intensive systems. In *Proceedings of ACM SIGMOD Conference* (Chicago, Ill., June 1988) ACM, NY, 1988.
2. Bitton, D. and Gray, J. Disk shadowing. In *Proceedings of the Fourteenth International Conference on Very Large Data Bases* (Los Angeles, Calif., August, 1988).
3. Boral, H. and DeWitt, D. Database machines: An idea whose time has passed? A critique of the future of database machines. In *Proceedings of the 1983 Workshop on Database Machines.* H.-O. Leilich and M. Missikoff, Eds., Springer-Verlag, 1983.
4. Boral, H. et al. Prototyping Bubba: A highly parallel database system. *IEEE Knowl. Data Eng. 2,* 1, (Mar. 1990).
5. Codd, E.F. A relational model of data for large shared databanks. *Commun. ACM 13,* 6 (June 1970).
6. Copeland, G., Alexander, W., Boughter, E., and Keller, T. Data placement in Bubba. In *Proceedings of ACM-SIGMOD International Conference on Management of Data* (Chicago, May 1988).
7. DeWitt, D.J., Katz, R., Olken, F., Shapiro, D., Stonebraker, M. and Wood, D. Implementation techniques for main memory database systems. In *Proceedings of the 1984 SIGMOD Conference,* (Boston, Mass., June, 1984).
8. DeWitt, D., et al. GAMMA—A high performance dataflow database machine. In *Proceedings of the 1986 VLDB Conference* (Japan, August 1986).
9. DeWitt, D., et al. The Gamma database machine project. *IEEE Knowl. Data Eng. 2,* 1 (Mar. 1990).
10. Engelbert, S, Gray, J., Kocher, T., and Stah, P. A benchmark of nonstop SQL Release 2 demonstrating near-linear speedup and scaleup on large databases. Tandem Computers, Technical Report 89.4, Tandem Part No. 27469, May 1989.
11. Ghandeharizadeh, S., and DeWitt, D.J. Performance analysis of alternative declustering strategies. In *Proceedings of the Sixth International Conference on Data Engineering* (Feb. 1990).
12. Ghandeharizadeh, S., and Dewitt, D.J. Hybrid-range partitioning strategy: A new declustering strategy for multiprocessor database machines. In *Proceedings of the Sixth International Conference on Very Large Data Bases,* (Melbourne, Australia, Aug. 1990).
13. Gibbs, J. Massively parallel systems, rethinking computing for business and science. *Oracle 6,* 1 (Dec. 1991).
14. Graefe, G. Encapsulation of parallelism in the Volcano query processing system. In *Proceedings of 1990*

ACM-SIGMOD International Conference on Management of Data (May 1990).

15. Graefe, G., and Ward, K. Dynamic query evaluation plans. In *Proceedings of the 1989 SIGMOD Conference,* (Portland, Ore., June 1989).
16. Hirano, M.S. et al. Architecture of SDC, the super database computer. In *Proceedings of JSPP '90.* 1990.
17. Hua, K.A. and Lee, C. Handling data skew in multiprocessor database computers using partition tuning. In *Proceedings of the Seventeenth International Conference on Very Large Data Bases.* (Barcelona, Spain, Sept. 1991).
18. Kitsuregawa, M., Tanaka, H., and Moto-oka, T. Application of hash to data base machine and its architecture. *New Generation Computing 1,* 1 (1983).
19. Kitsuregawa, M., Yang, W., and Fushimi, S. Evaluation of 18-stage pipeline hardware sorter. In *Proceedings of the Third International Conference on Data Engineering* (Feb. 1987).
20. Kitsuregawa, M., and Ogawa, Y. A new parallel hash join method with robustness for data skew in super database computer (SDC). In *Proceedings of the Sixteenth International Conference on Very Large Data Bases.* (Melbourne, Australia, Aug. 1990).
21. Lorie, R., Daudenarde, J., Hallmark, G., Stamos, J., and Young, H. Adding intra-transaction parallelism to an existing DBMS: Early experience. *IEEE Data Engineering Newsletter 12,* 1 (Mar. 1989).
22. Patterson, D. A., Gibson, G. and Katz, R. H. A case for redundant arrays of inexpensive disks (RAID). In *Proceedings of the ACM-SIGMOD International Conference on Management of Data.* (Chicago, May 1988).
23. Ries, D. and Epstein, R. Evaluation of distribution criteria for distributed database systems. UBC/ERL Technical Report M78/22, UC Berkeley, May, 1978.
24. Salem, K. and Garcia-Molina, H. Disk-striping. Department of Computer Science, Princeton University Technical Report EEDS-TR-322-84, Princeton, N.J., Dec. 1984.
25. Schneider, D. and DeWitt, D. A performance evaluation of four parallel join algorithms in a shared-nothing multiprocessor environment. In *Proceedings of the 1989 SIGMOD Conference* (Portland, Ore., June 1989).
26. Schneider, D. and DeWitt, D. Tradeoffs in processing complex join queries via hashing in multiprocessor database machines. In *Proceedings of the Sixteenth International Conference on Very Large Data Bases.* (Melbourne, Australia, Aug., 1990).
27. Selinger P. G., et al. Access path selection in a relational database management system. In *Proceedings of the 1979 SIGMOD Conference* (Boston, Mass., May 1979).
28. Stonebraker, M. Muffin: A distributed database machine. ERL Technical Report UCB/ERL M79/28, University of California at Berkeley, May 1979.
29. Stonebraker, M. The case for shared nothing. *Database Eng. 9,* 1 (1986).
30. Stonebraker, M., Katz, R., Patterson, D., and Ousterhout, J. The design of XPRS. In *Proceedings of the Fourteenth International Conference on Very Large Data Bases.* (Los Angeles, Calif., Aug. 1988).
31. Tandem Database Group. NonStop SQL, a distributed, high-performance, high-reliability implementation of SQL. Workshop on High Performance Transaction Systems, Asilomar, CA, Sept. 1987.
32. Tandem Performance Group. A benchmark of non-stop SQL on the debit credit transaction. In *Proceedings of the 1988 SIGMOD Conference* (Chicago, Ill., June 1988).
33. Teradata Corporation. DBC/1012 Data Base Computer Concepts & Facilities. Document No. C02-0001-00, 1983.
34. Tevanian, A., et al. A Unix interface for shared memory and memory mapped files under Mach. Dept. of Computer Science Technical Report, Carnegie Mellon University, July, 1987.
35. Thakkar, S.S. and Sweiger, M. Performance of an OLTP application on symmetry multiprocessor system. In *Proceedings of the Seventeenth Annual International Symposium on Computer Architecture.* (Seattle, Wash., May, 1990).
36. *The Performance Handbook for Database and Transaction Processing Systems.* J. Gray, Ed., Morgan Kaufmann, San Mateo, Ca., 1991.
37. Walton, C.B., Dale, A.G., and Jenevein, R.M. A taxonomy and performance model of data skew effects in parallel joins. In *Proceedings of the Seventeenth International Conference on Very Large Data Bases.* (Barcelona, Spain, Sept. 1991).
38. Wolf, J.L., Dias, D.M., and Yu, P.S. An effective algorithm for parallelizing sort-merge joins in the presence of data skew. In *Proceedings of the Second International Symposium on Parallel and Distributed Systems.* (Dublin, Ireland, July, 1990).
39. Zeller, H.J. and Gray, J. Adaptive hash joins for a multiprogramming environment. In *Proceedings of the 1990 VLDB Conference* (Australia, Aug. 1990).

CR Categories and Subject Descriptors: B.5.1 [**Register-Transfer-Level Implementation**]: Design-style (e.g., parallel, pipelined, special-purpose); C.1.2 [**Computer Systems Organization**]: Processor Architectures—*Multiple Data Stream Architectures (Multiprocessors)*; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*Parallelism*; H.2.1 [**Information Systems**]: Database Management—*Logical design*; H.2.8 [**Information Systems**]: Database Management—*Database Applications*; H.3 [**Information Systems**]: Information Storage and Retrieval

General Terms: Design, Measurement

Additional Keywords and Phrases: Parallelism, parallel database systems, parallel processing systems.

About the Authors:

DAVID DEWITT is a professor in the Computer Sciences Department at the University of Wisconsin. His current research interests include parallel database systems, object-oriented database systems, and database performance evaluation. **Author's Present Address:** Computer Sciences Department, University of Wisconsin, 1210 West Dayton Street, Madison, WI 53706; email: dewitt@cs.wisc.edu

JIM GRAY is a staff member with the Digital Equipment Corporation. His current research interests include databases, transaction processing, and computer architecture. **Author's Present Address:** San Francisco Systems Center, Digital Equipment Corporation, 455 Market Street—7th Floor, San Francisco, CA 94105-2403; email: gray@sfbay.enet.dec.com

This research was partially supported by the Defense Advanced Research Projects Agency under contract N00039-86-C-0578, by the National Science Foundation under grant DCR-8512862, and by research grants from Digital Equipment Corporation, IBM, NCR, Tandem, and Intel Scientific Computers.

An Object-Oriented Methodology for Knowledge Base/Database Coupling

Kunihiko Higa

Mike Morrison

Joline Morrison

Olivia R. Liu Sheng

Databases have evolved into the central component of organizational information systems over the past two decades. However, the typical database lacks the semantic support needed for exception handling, query optimization, and some update constraints. A knowledge base can provide this for a database; specifically, separating abstract knowledge into a knowledge base and facts into a database enhances the maintenance and portability of both and thereby increases their life spans [1, 15, 22]. Advantages of coupled knowledge base/database (KB/DB) systems have been widely recognized by both researchers and practitioners [1, 10, 15, 21, 40]. Unfortunately, large knowledge bases (such as those required to support database semantics) are difficult to develop and maintain because of their limited capacity for handling large amounts of factual data. Our goal is to investigate how to achieve a natural and effective KB/DB coupling.

Intensional knowledge has been classified as "knowledge beyond the factual content of the database" [45]. Intensional knowledge is more abstract than extensional knowledge items and the system knowledge needed for query and response control. The least abstract subcategory of intensional knowledge, *structural knowledge*, comprises the

structural dependencies and constraints among data. The purpose of this article is to present a generalizable methodology for representing structural knowledge implicit to a database in a knowledge base to form an integrated knowledge base/database system. This will be done using the Structured Object Method (SOM), an object-oriented graphical model developed for database and knowledge base design [19, 20]. The SOM is useful for coupled KB/DB system design because its tree structure aids in representing the database schema information in a way that lends itself to the development of knowledge-based search and inferencing strategies. Example implementation strategies using both an object-oriented development environment and an expert system shell will be described.

We first describe our research environment and motivation for developing a coupled KB/DB system. Related work in the areas of coupled knowledge base/database systems and object-oriented coupled system design is then surveyed to show the absence of an existing coupled systems methodology and to investigate potential foundations for our methodology. Our coupled KB/DB methodology is then described and illustrated using both an object-oriented development environment and an object-oriented expert system shell. Related research and contributions of this study are presented in the final section of this article.

Research Environment and Motivation

The Integrated Office Information System (IOIS) project was an ongoing project within the University of Arizona's Management Information Systems Department through 1989. The main objective of the IOIS project [24] was to develop an environment where diverse office information systems containing data and knowledge could be integrated. By allowing access to data and knowledge through a common interface, the IOIS can potentially improve office productivity.

To develop a prototype system, a KB/DB requirements analysis based upon a case study of the IOIS funding agency was used [33]. The KB requirements include procedures, policies, organizational structures, and other knowledge pertinent to the office. This knowledge is characterized by a high rule-to-fact ratio: i.e., the facts are highly interrelated. Less interrelated facts, essential to the office's data processing, are stored in the database. All data and knowledge used by the components of the IOIS are to be provided by its database and knowledge base.

Our goal is to use the KB as a front end to support more sophisticated queries and updates (e.g., query navigation, data integrity maintenance, etc.) than those allowed by conventional database management systems. For example, if an office has different purchase order forms and order procedures for different equipment, the KB can identify the appropriate form and procedure for a particular piece of equipment. Because the actual order form and procedure are stored in the DB, the KB must access the DB to instantiate the form and procedure.

To proceed from requirements analysis to design and eventual implementation of a prototype system, a methodology for developing coupled KB/DB systems must be identified or developed. Our first step is to review related work.

Related Work

Previous work on development of coupled KB/DB systems can be divided into the areas of *direct data-retrieval systems* (systems that provide direct access to data sets) and *document retrieval systems* (systems that provide only indications of data set contents) [31]. Research involving both types of systems has used traditional knowledge representation techniques such as predicate logic [3], production rules [7, 12], hierarchically-structured rules [27], semantic nets [11, 46], and frames [5, 13, 34]. Representation techniques using combinations of these techniques have also been developed: *RUBRIC* [26, 44] used production rules in hierarchical tree structure; *CoalSORT* [29] employed a frame-based semantic net; *EP-X* [42] used a "tangled" (i.e., network-like) hierarchy; and, *CANSEARCH* [35] utilized a hierarchical knowledge organization combined with an abstract search space limited to the relevant domain to aid search efficiency.

Many coupled KB/DB-related efforts have been aimed toward knowledge modeling; for example, a knowledge base development model with object-oriented properties based on the Semantic Data Model [17] was developed to capture knowledge and data semantics [36]; a formalism for describing semantic databases that provides a foundation for developing a coupled knowledge base is presented in [39]. However, neither of these efforts described a development methodology.

Other work is more implementation-oriented, but again, specific methodologies are not developed. Leung and Wong [23] apply the object-oriented paradigm to expert system design to develop an object-oriented expert system architecture and suggest that this architecture could be used for coupled KB/DB systems, but do not specify object structures or methods for representing database structural knowledge or developing queries. Gorman and Choobineh [16] present a description of the Object-Oriented Entity-Relationship Model (OOERM), an object-oriented extension to the Entity-Relationship Model [6] for modeling entity class dynamics. The associated query language includes syntax for database queries, but the relationships between the knowledge base and database and the query development mechanics are not specified. Ram, Hayne and Carlson [37] describe a coupled system implementation with an object-oriented knowledge base architecture using

a hierarchical tree structure, but their work is domain-specific and does not attempt to describe a general implementation procedure. We conclude that no methodologies for developing and representing structural database knowledge have been proposed. Furthermore, no known coupled KB/DB systems using object-oriented development principles have used a hierarchical object structure.

We have chosen an object-oriented strategy for our KB/DB coupling methodology. Leung and Wong [23] suggest that traditional knowledge representation techniques are not structured enough to effectively manage large knowledge base systems (such as those required to represent the structural knowledge associated with a database); they recommend using an object-oriented representation to enhance knowledge base structure, modularity, and maintenance. Additionally, the object-oriented approach is well suited to modeling data and database relationships since it is easily correlated to well-known conceptual data models such as the ER Model and the Semantic Data Model [2]; and, the object-oriented paradigm has evolved into accepted software development technology with recognized benefits in the areas of software maintainability, extensibility, and reusability [28]. A hierarchical structure will be applied to this object-oriented strategy to provide a blueprint for relating objects and applying search and inference strategies using inheritance. We propose to use Pollitt's [35] approach of developing abstract limited search spaces to increase system efficiency. The methodology will be described in the following section.

KB/DB Coupling Methodology

The Structured Object Model (SOM) [19] was used for data and knowledge representation because it provides the desired hierarchical and object-oriented representation environment. We will first describe SOM, and then present the SOM-based coupled KB/DB development methodology.

The Structured Object Model

The Structured Object Model (SOM) method is a structured methodology derived from the System Entity Structure [48, 49]. SOM represents data semantics using objects, attributes, and two types of relationships: *aspects* and *specializations*. SOM can be used for logical and physical database design, knowledge base design, and coupled KB/DB system design. SOM will be discussed in terms of general object-oriented concepts and its means for representing data relationships. Comparisons will be drawn between SOM and other database representation techniques used for database and knowledge base design.

Object-Oriented Concepts.

From a data modeling perspective, an object may be a person, place, thing, organization, concept, or event about which users wish to collect and store information [9, 38]. Sets of objects exhibiting common structures and behaviors are grouped into classes [4]. In SOM, tangible object types with multiple instances (such as projects, employees, and equipment) are defined as classes, with each instantiation representing a separate object.

Class and object representations are formed by constants and variables with either common class values or individual instance values (e.g., *class* and *instance variables* in Smalltalk; *member objects* in C++). Similarly, SOM uses *attributes* to identify and describe objects by providing them properties such as name, shape, and color. There are two types of SOM attributes: **identifiers** and **descriptors.** An identifier value (i.e., key) uniquely identifies an object instance, and a descriptor describes the state or properties of the instance.

Object-oriented methods are something like conventional procedure calls, except that concepts such as polymorphism, inheritance, and dynamic binding allow them to be more powerful and versatile [4]. SOM also supports methods in the form of formulas, rules, and procedures that can be applied to classes, subsets of classes, or individual data objects.

SOM Data Relationships.

An SOM diagram is constructed by decomposition and specialization. Decompositions represent object components, while specializations represent object classifications. An *aspect* relationship occurs when an object is decomposed into subobjects that represent components of the parent. For example, aspects of a car would include the engine, wheels, and steering wheel. A *specialization* relationship indicates that a parent object can be specialized in only one of its subobjects. Specializations of a car would be sedans, wagons, and vans.

Comparisons Between SOM and other Data Representation Techniques.

The foremost conceptual data models in use today include the Entity-Relationship Model (ER Model) and its derivatives [6, 8] and the Semantic Data Model (SDM) [17]. The strengths and weaknesses of SOM compared to these models are discussed in the following two paragraphs.

SOM and the ER Model. The ER Model shows data entity relationships graphically, where entities are nodes and relationships are edges. The ER Model is more useful than SOM for identifying data relationships during analysis and for showing ternary relationships [30]. However, the ER Model cannot be used to show hierarchical relationships, so SOM provides an extra component of semantic data. Furthermore, for complex systems with many interrelated edges, SOM can be easier to understand and interpret, and SOM presents a superior model for representing structural database relationships in knowledge bases [30].

SOM and the SDM. The SDM uses a textual class-object syntax to describe data and data relationships.

This approach provides a richer level of semantic data than visual representations such as SOM, but users find nonhierarchical representations (like SDM) harder to assimilate and understand [41]. Furthermore, the SDM syntax is complex for novice users to learn and use, while SOM is easily learned and applied by novices [25]. Again, SOM provides a model for structural database knowledge that is more easily translated to knowledge base representations.

Coupled KB/DB Development Methodology

Our coupled KB/DB methodology

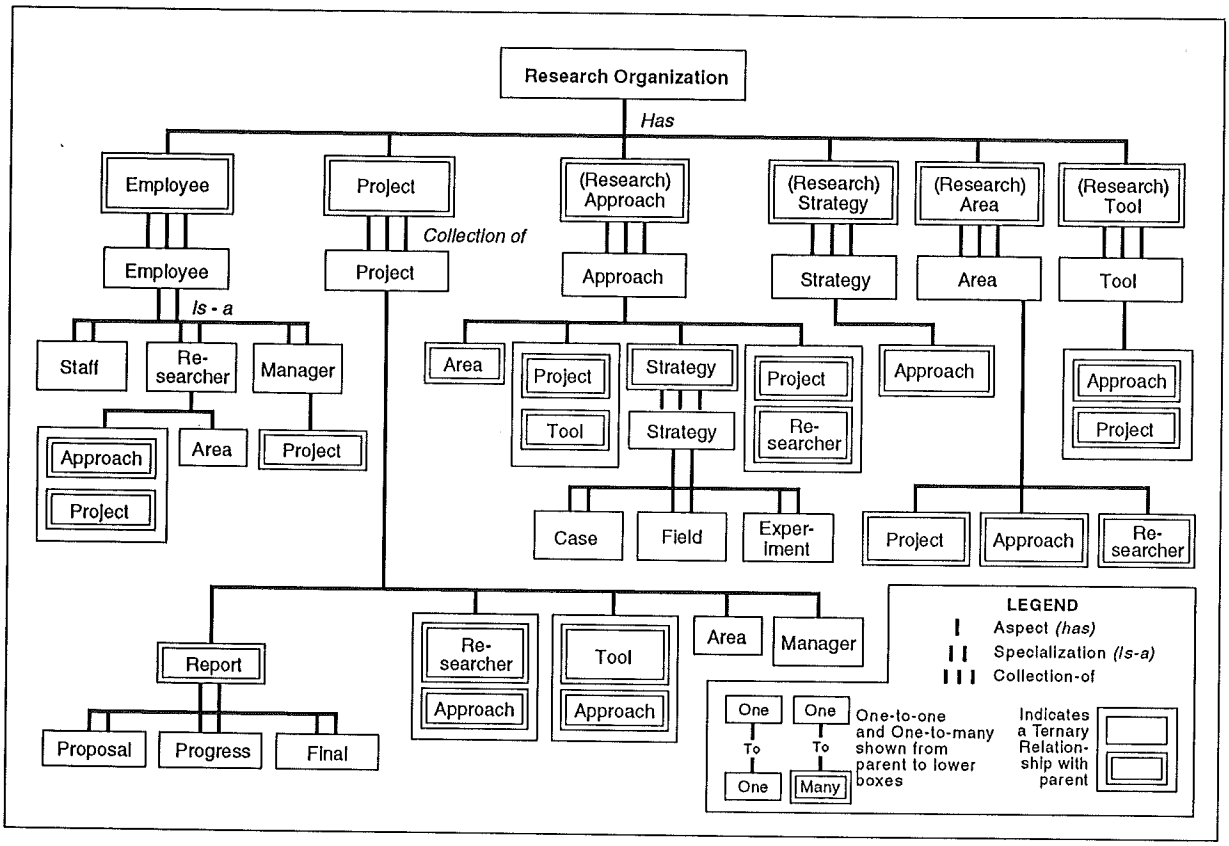


Figure 1. Logical SOM Diagram

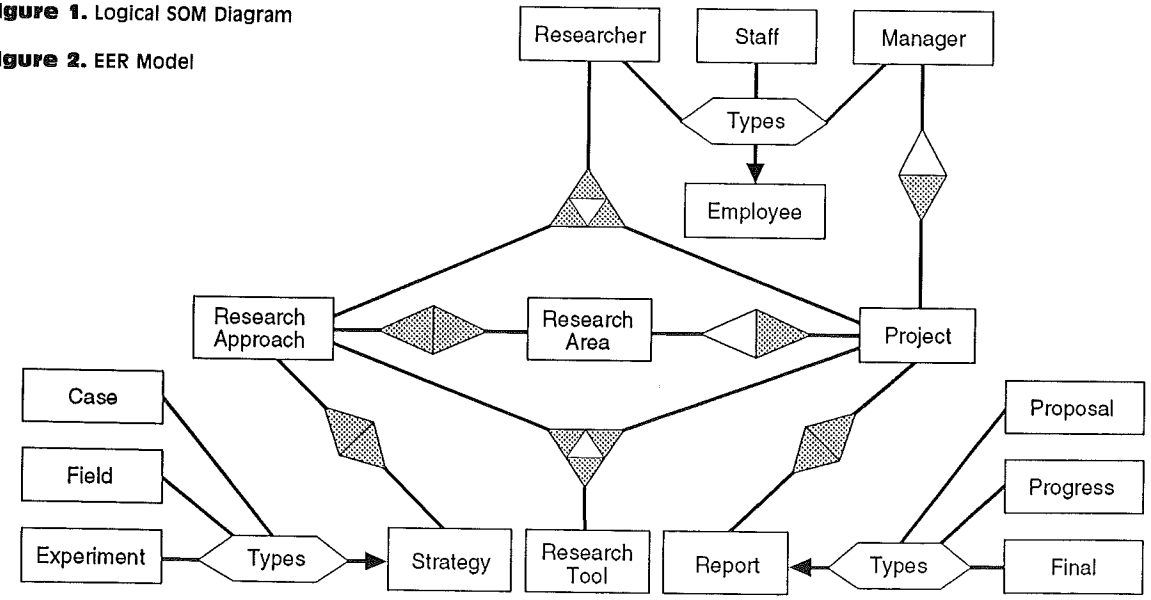


Figure 2. EER Model

first develops the logical and physical database schema for the target system, and then uses this design to identify object relationships that will distinguish knowledge subsets, search paths, and inferencing patterns that are then represented in the knowledge base. We will first describe the SOM database development procedure, and then build upon this design to identify the steps involved in coupled system knowledge base design.

Database Development.

To illustrate the SOM database development process, we will describe the design procedure for a subset of the database identified in the database requirements analysis study described previously in the section "Research Environment and Motivation." The following steps were used:

Step 1: Define the most general object and identify all relevant classes and attributes. The general object is *Research Organization.* The relevant objects, descriptions, and attributes are as follows:

1. EMPLOYEE describes research organization personnel. Specializations include *Researcher, Staff,* and *Manager;* attributes include *SSN, Name, Title, Department, Research Area.*
2. PROJECT describes research projects; an instance of this class provides information about a specific research project. Attributes include *PID (Project ID), Title, Manager, Research Area, Contractor, Start Date, Stop Date.*
3. REPORT describes report types such as PROPOSAL, PROGRESS, and FINAL. Attributes include *Report ID* and *Report Type.*
4. RESEARCH APPROACH describes a project's RESEARCH AREA, STRATEGY, TOOL, and RESEARCHERS.
5. RESEARCH STRATEGY describes various research strategies used in the research organization (e.g., case study, prototype, experiment, etc.). Attributes include *Strategy Name.*
6. RESEARCH AREA describes research areas investigated by the organization, such as office automation, data communications, and information storage and retrieval. Attributes include *Area Name, Area Description, Area Keywords.*
7. RESEARCH TOOL includes both hardware and software research tools used in the organization. Attributes include *Tool Name, Tool Description, Tool Keywords.*

Step 2: Define relationships between classes. Our experience indicates that Steps 1 and 2 should be done iteratively using SOM with another conceptual database model such as the ER Model or Extended Relational Model (EERM) [8]. These network-based models are superior for identifying ternary entity relationships, but are not as well-suited as SOM for representing structural knowledge or recognizing inferencing patterns.

The results of Steps 1 and 2 are the Logical SOM and EERM diagrams shown in Figure 1 and Figure 2. The SOM shows entity classes EMPLOYEE, PROJECT, APPROACH, STRATEGY, AREA, and TOOL. The REPORT class is shown on the second level as a component of PROJECT.

Relationships illustrated on the Logical SOM will be discussed using the PROJECT component, which has the following aspects: REPORT, RESEARCHER/APPROACH (a ternary relationship whereby a single project may have multiple researchers using multiple approaches), TOOL/APPROACH (a similar ternary relationship), AREA, and MANAGER. REPORTS are specialized into types of reports: progress, proposal, and final report. These relationships are also illustrated on the EER.

Step 3: Check completeness of relationships. To ensure that no incomplete binary or ternary relationships exist on the Logical SOM, all binary relationships should appear twice on the diagram, and all ternary relationships should appear three times. For example, under the top-level EMPLOYEE class, the binary relationship of RESEARCHER to AREA is shown. A similar relationship exists under the AREA class, where RESEARCHER is shown as a component.

Step 4: Simplify the SOM diagram and copy each object structure to a relation. A set of rules have been developed to simplify the Logical SOM into a set of relations representing the physical database design. The result of applying these rules to our Logical SOM example is shown in Figure 3.

1. ***One-to-one relationships:*** *Copy each other's identifier to a single object structure, then delete the portion of the diagram which represents the one-to-one relationship.*

No relationships of this type exist in Figure 3.

2. ***One-to-many relationships:*** *Copy the one-side's identifier into the many-side's identifier, and then delete the portion of the diagram that represents the one-to-many relationship.*

An example of this operation is in the relationship between PROJECT and AREA. Project instances have a single research area, but a research area may involve many projects. Therefore, AREA's identifier, Area_Name, becomes an attribute within the PROJECT object structure.

3. ***Many-to-many relationships:*** *Create a new object by copying identifiers of the two objects; both identifiers form a composite key for the new object. Delete the portions of the diagram that identify the many-to-many relationship, and attach the new object to the original objects.*

This operation is illustrated in the relationship between AREA and APPROACH. A new object, APPROACH-AREA, is created, with its identifier composed of both Approach-Name and Area-Name.

4. ***Ternary relationships:*** *Create a new object by copying identifiers of all three objects; these three identifiers form the composite key for the new object.*

The relationship between PROJECT, TOOL, and APPROACH illustrates this operation. A new object, PROJECT-TOOL-APPROACH, is created, with com-

posite identifiers PID, Tool-Name, and Approach-Name.

When the SOM structure is completed, the relation corresponding to each object in the structure is in at least third normal form [19]. Every object in Figure 3 (PERSON, PROJECT, RESEARCH TOOL, etc.) contains its own set of attributes and is in third normal form.

Five of the original seven objects identified in Step 1 of the database development (EMPLOYEE, PROJECT, APPROACH, AREA, and TOOL) are directly translated into relations, and are thus described as *simple objects*. As a result of simplifying the SOM, the following new, or *complex objects*, are created:

- EMPLOYEE-PROJECT-APPROACH, which specifies (1) all employees who work on a particular project; (2) all projects that a given employee is involved with; (3) all approaches that a given project uses; (4) all approaches that a given employee is connected with; and (5) the unique approach used by a given employee on a given project.
- PROJECT-TOOL-APPROACH, which specifies (1) all projects using a given approach; (2) all approaches used on a given project; (3) all tools used on a given project; (4) all tools used with a given approach; and (5) the unique tool associated with a given approach on a given project.
- PROJECT-REPORT, which describes report characteristics for given projects.
- APPROACH-STRATEGY, which specifies research approaches for a given strategy and strategies for a given approach.
- APPROACH-AREA, which describes research approaches for a given research area, and appropriate research areas for a given approach.

The schema described above (for both simple and complex objects) and the rest of the IOIS DB schema were developed using this development method. The entire prototype database therefore is consistent and has a natural association with the prototype knowledge base, to be developed in the following subsection.

Knowledge Base Development.

In general, there are two types of knowledge: extensional (episodic), and intensional (semantic). In the context of a database, extensional

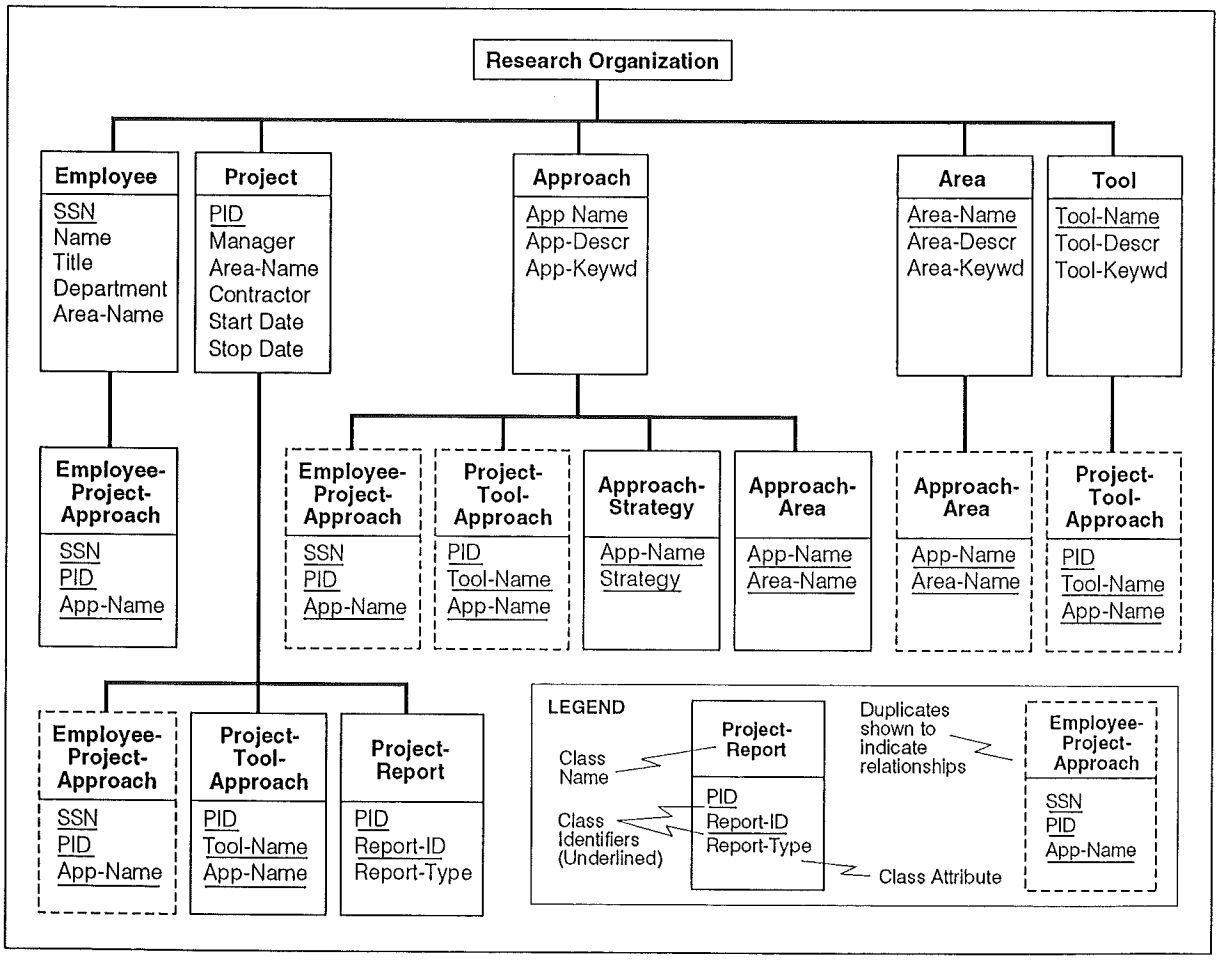


Figure 3. Physical SOM

knowledge consists of instantiations of organizational events and objects. These are stored in a database. Intensional knowledge describes the database's structure, and can be further categorized into general knowledge (analogous to a global view of a database) and task-specific knowledge (analogous to an external or user view of a database). General and task-specific semantic knowledge base components are described in the following paragraphs, and a methodology for implementing task-specific semantic knowledge is developed.

General Semantic Knowledge. General semantic knowledge (GSK) includes descriptions of organizational objects and associations among those objects. In fact, the Physical SOM diagram is GSK. Although this can be extended by including knowledge regarding topics such as organizational security measures and data retrieval/update policies to allow GSK to be used as a security guard and policy enforcer of the organizational DB. The design of GSK must precede the design of task-specific semantic knowledge because the latter uses components of the former.

Task-Specific Semantic Knowledge. Information retrieval tasks typically involve multiple entities or objects. These objects are organized to form a specific structure on which searching or inferencing may be performed to accomplish a desired task. After the general knowledge structure is constructed, substructures may be extracted and/or coupled together to represent specialized knowledge structures for specific tasks. Each specialized knowledge structure is a subset (or chunk) of the knowledge base. This newly formed structure, together with its search strategies or inferencing patterns, is an example of Task-Specific Semantic Knowledge. TSK is developed by first collecting required objects for the task from the GSK structure, and then creating a specific association among these objects to form an abstracted SOM view for the task. The designer then defines search strategies or inference patterns. Specifically, design steps are as follows:

Step 1: Develop a representation of the database structure using object-oriented principles to show data relationships. This is accomplished by implementing each SOM class as an object class in the given development environment. Class attributes are represented as class variables, and relationships may be represented through inheritance, as well as using (aspect) and instantiation (specialization) relationships. (For an overview of these relationships, see [4]).

Step 2: Collect required objects from GSK and construct an abstracted SOM diagram based on the desired application. An abstracted SOM view is created by analyzing implementation inputs and outputs, and determining an exhaustive set of related database objects. This may be accomplished either manually or through knowledge-based techniques.

Step 3: Define search methods or strategies. Structural database components in the abstracted SOM may be instantiated through three ways: (1) user inputs; (2) retrieved database data; and (3) data relationships. Methods and strategies can be developed by analyzing relationships between given inputs, desired outputs, and structural database knowledge. When all required objects in the knowledge structure have been instantiated through searching or inferencing, the desired output(s) are identified.

Example design and implementation sequences using both an object-oriented development environment and expert system shell will be described in the following section.

Methodology Implementation Examples

To demonstrate this methodology, the following task relevant to the previously described IOIS database will be used: let us suppose that to define a new project specification, a user needs to retrieve all project reports containing certain keywords. Keywords may be related to research strategy, research area name, and/or research tool. Example implementations of this application will be demonstrated using *PC Scheme* [43], a LISP dialect with an object-oriented extension, *SCOOPS (Scheme Object-Oriented Programming System),* and *Nexpert Object* [32], an object-oriented expert system shell. Based on these examples, strengths and weaknesses of the proposed methodology will be discussed in the following subsections.

PC Scheme Implementation

PC Scheme is an IBM-compatible PC-based *LISP* dialect with an object-oriented environment that supports most major features of the object-oriented model. The example implementation sequence is as follows:

Step 1: Develop a representation of the database structure using object-oriented principles to show data relationships. The database class-object structure was defined according to Physical SOM (Figure 3). An object-oriented database requires iterative methods to sequentially access class members. Since containing-class relationships are not supported in *SCOOPS,* we explicitly defined our inheritance relationships by first defining a class and then immediately creating a "containing" instance with instance variable *child-list* to identify hierarchical successors. Specifically, the class RESEARCH-ORG was defined, and then the instance *research-org-obj* was defined with its *child-list* composed of the names of its successors: EMPLOYEE, PROJECT, APPROACH, AREA, and TOOL. The rest of the containing-class instances were created, with leaf classes (such as PROJECT-REPORT) having null-valued child-lists. This representation provides the structural relationships of the database.

To facilitate searching for required identifiers or attributes, each containing object must have knowledge of its own identifiers and attributes, as well as knowledge

about the attributes of related objects. This was accomplished by creating a structure in each object consisting of the names of each of its attributes, as well as a corresponding list of the names of other database containing objects with the same attribute. From this knowledge, abstracted SOM diagrams can be derived.

Syntax for the structure definition of the containing object TOOL is shown in Figure 4. All attributes as well as the names of corresponding objects with the same attributes are held in a list (attribute-list) composed of structures called *attributes,* and consisting of both the attribute name and its corresponding object locations. The order in which these attributes and their locations are placed in the object (and ultimately searched) is a design decision based on characteristics of the application and data. For example, highly interrelated data may benefit from a depth-first search, since related objects would be searched first, and less dead ends from searching unrelated objects would result. Conversely, data with few interrelationships could benefit from a breadth-first search since the desired attributes and locations are probably in unrelated objects.

Step 2: Collect required objects from GSK and construct an ab-

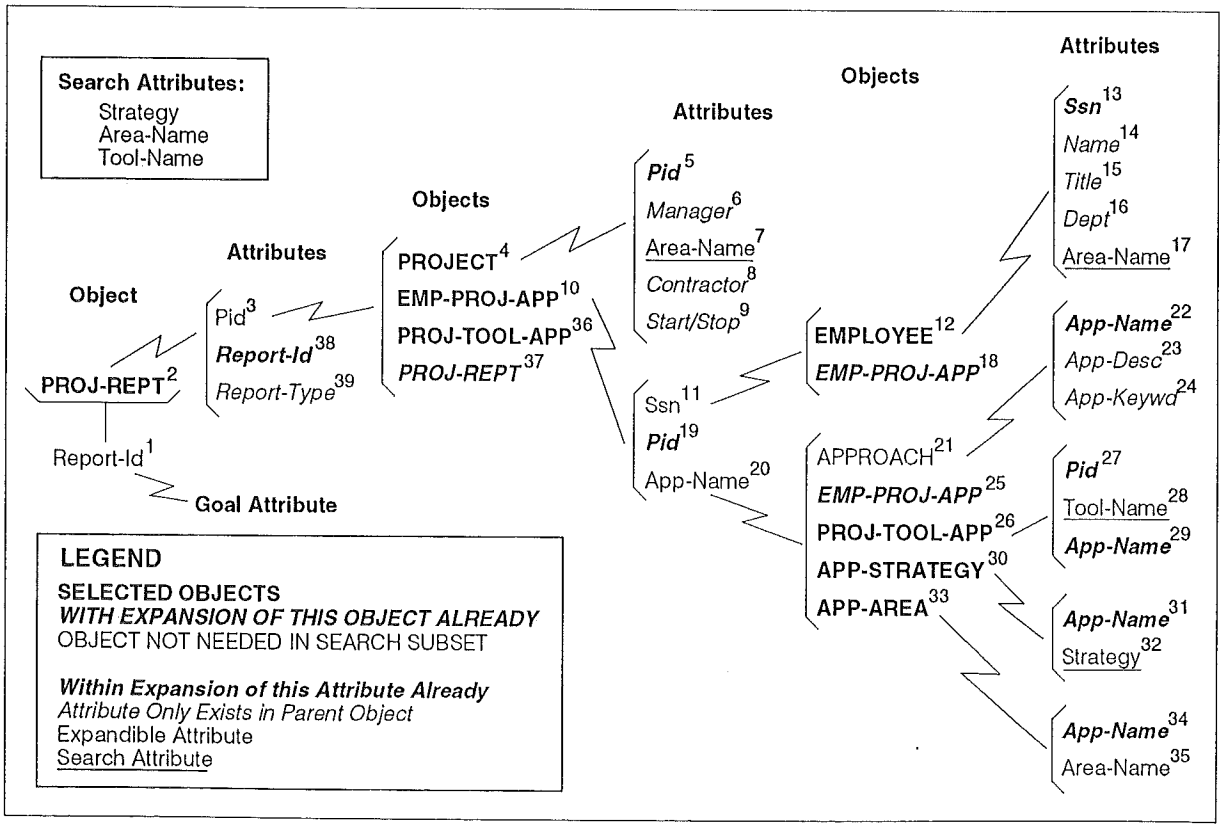
```
(define tool
    (make-instance research-org
        'child-list        '(project-tool-approach)
        'identifiers       '(tool-name)
        'identifier-locs   '(project-tool-approach)
        'attribute-list    (list    (eval '(define (make-attributes
                                        'name  'tool-name
                                        'locs  '(project-tool-approach)
                                    (eval '(define (make-attributes
                                        'name  'tool-descr
                                        'locs  '())
                                    (eval '(define (make-attributes
                                        'name  'tool-keywds
                                        'locs  '()))))
```

Figure 4. Syntax for the Structure Definition of the Containing Object TOOL

```
(define-method (research-org ask-all)(exp)
    (make-instance research-org 'members
        (map (lambda(el)
            (send-if-handles el eval-self exp)) members)))
```

Figure 5. Syntax for Container-Object "Members"

Figure 6. Search Procedure to Derive Abstracted SOM



stracted SOM diagram. Objects containing goal attribute(s) must first be determined. To do this, all objects must be accessed to determine if the given goals are included within their attributes. This is accomplished using a method developed by Zeigler [50] for parallel searching of container object members (but applicable to sequential searching as well). This method, *ask-all,* queries all member objects of an invoked object regarding a specified condition. The method outcome is a newly created container-object called "members" containing all of the member objects who evaluated the condition as true. The syntax is as shown in Figure 5. This method would be invoked upon all database containing objects to find the given goal (REPORT-ID) in a recursive, depth-first search, with each object invoking the method on its child-list (excepting those previously searched through another containing object).

Next, objects related to the goal through the input parameters must be identified. This search procedure is illustrated in Figure 6. Starting from the goal object (PROJECT-REPORT), the first step is to determine those objects that can instantiate the value of the required identifier (Report-ID). This is done through a backward-chaining search strategy as follows: attribute values that could potentially instantiate values for Report-ID in PROJECT-REPORT include PID and Report-Type. The identities of other objects that could potentially instantiate these attributes are encapsulated in the PROJECT-REPORT object; PID has several related object locations, while Project-Type is a local attribute with no other locations. The first PID location (PROJECT) is accessed, and it is determined that one of the input parameters (Research-Area-Name) is an attribute in this object also. Therefore, PROJECT is a related object for this query (since the value of PID can be instantiated through the input parameter Research-Area-Name) and will become part of the abstracted SOM

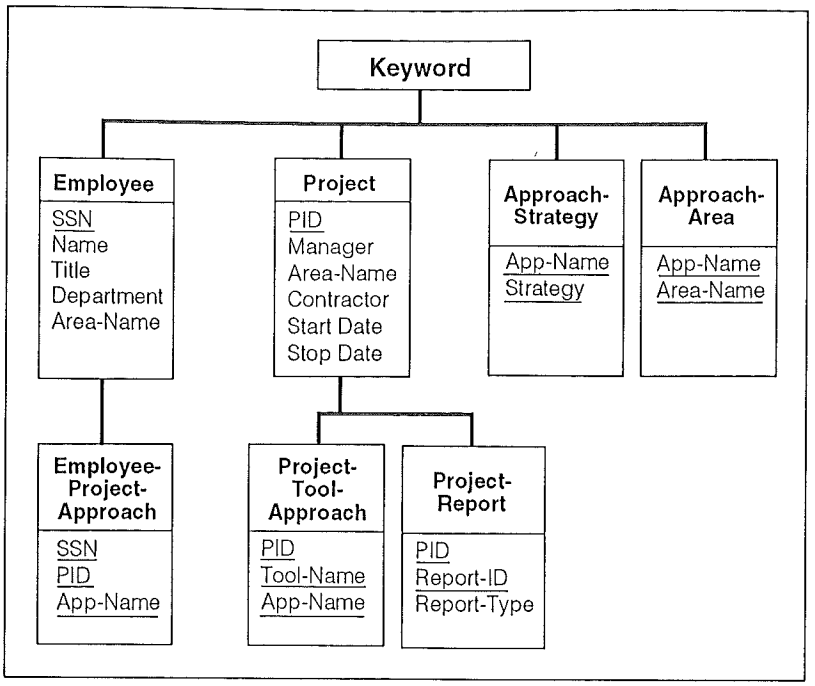


Figure 7. Abstracted SOM View

diagram. During the actual query process, user-specified relationships between the input parameter being used to instantiate the current value versus the other input parameters (e.g., Strategy and Area-Name and Tool-Name versus Strategy or Area-Name or Tool-Name) will determine if PIDs instantiated through the PROJECT object are joined as unions or manipulated as intersections with PIDs instantiated through other input parameters.

The next PID location (EMPLOYEE-PROJECT-APPROACH) is then accessed; repeating the search algorithm, it is discovered that Research-Area-Name can instantiate PID through the EMPLOYEE object. Therefore, both EMPLOYEE and EMPLOYEE-PROJECT-APPROACH are added to the containing object for the abstracted SOM. This search continues until all related objects are found. The required objects for this application are identified as: PROJECT-REPORT—contains the goal; EMPLOYEE—contains a relationship between research area and Project ID, needed to select reports; EMPLOYEE-PROJECT-APPROACH—defines a relationship between Project ID and research area through research approach; PROJECT—defines a relationship between Project ID and research area; PROJECT-TOOL-APPROACH—defines a relationship between Project ID, tool name, and research area (through the attribute *research approach*); APPROACH-STRATEGY—contains the relationship between approaches and strategies; and, APPROACH-AREA—contains the relationship between approaches and areas. This abstracted SOM view is shown in Figure 7.

To effectively use this abstracted SOM object set, copies of all selected objects must be created and modified to reflect the reduced search space. To do this, attribute locations (and the corresponding attributes) outside of the reduced search space are deleted from each object's attribute list. This collection of modified objects then represents the Abstracted SOM for the particular application.

Step 3: Define search methods or strategies based on the SOM diagram and the development environment. The same search strategy

described for finding the abstracted SOM objects can be used for finding particular data instances. Once the search path from the given input(s) to the desired output(s) is derived, the path is traced in reverse. Each containing set is sequentially queried with the appropriate inputs; result sets are manipulated either as unions or intersections depending on user input specifications.

Nexpert Implementation

Nexpert connects class/object structures and rules by their common data, classes, or hypotheses, and thus forms object and rule networks. Both upward and downward inheritance is allowed between both classes and objects. Inferencing support includes nonmonotonic reasoning as well as forward and backward chaining. Each mechanism can be globally and locally controlled. The following sequence was followed to implement the given application:

Step 1: Develop a representation of the database structure using object-oriented principles to show data relationships. A database schema can be represented within a *Nexpert* KB using its class/object-net combined with its rule-net. SOM entities can be directly transformed into classes; relationships between SOM objects (e.g., many-to-many, many-to-one, and one-to-one) are then translated into rules.

An aspect class in SOM is directly transformed to a *Nexpert* class as shown in Figure 8a. In this case, the class names and attributes become *Nexpert* class names and properties respectively. When a class is the child of a specialization, the name of this class must be included as a subclass of the generalized class (see Figure 8b). Those classes are easily instantiated through data retrieval from the database because the *Nexpert* class definitions and DB schema are defined and coupled through the same SOM diagram.

Step 2: Collect required objects from GSK and construct an abstracted SOM diagram. A forward-chaining rule-net may be constructed to perform the following procedure:

1. *Extract the goal object that contains the goal attribute(s) with its parent object and its child structure.* Figure 3 shows PROJECT-REPORT to be the goal object that contains Report-ID, the goal attribute. PROJECT is the parent of the goal object, and the goal object does not have any children; therefore, only two objects, PROJECT and PROJECT-REPORT, are extracted.
2. *Extract all children of the parent object.* Figure 3 shows EMPLOYEE-PROJECT-APPROACH and PROJECT-TOOL-APPROACH to be the remaining children of the parent object PROJECT, and they are extracted.
3. *For all extracted objects that are complex objects (i.e., intersection records), extract all objects involved in the complex object.* Of the objects thus far extended, EMPLOYEE-PROJECT-APPROACH and PROJECT-TOOL-APPROACH are complex objects. Therefore, their associated objects (EMPLOYEE, TOOL, and APPROACH) objects are extracted. (Note that PROJECT was already extracted in 1.)
4. *Extract the children of every object extracted in the previous step only if a child object contains more explicit or implied input parameters than its parent, and delete the parent object if any of its children are extracted.* Figure 3 shows APPROACH to contain only one input parameter (Approach-Name). Its two children each contain both an explicit input parameter and an implicit input parameter: APPROACH-STRATEGY contains Strategy and Approach-Name, and APPROACH-AREA, contains Area-Name and Approach-Name. Therefore, both objects are extracted and their parent (APPROACH) is deleted.
5. *Extract children of objects extracted in 4, and repeat Steps 3 and 4 until no further extraction is possible.* This completes the abstracted SOM diagram as shown in Figure 7.

Step 3: Define search methods or

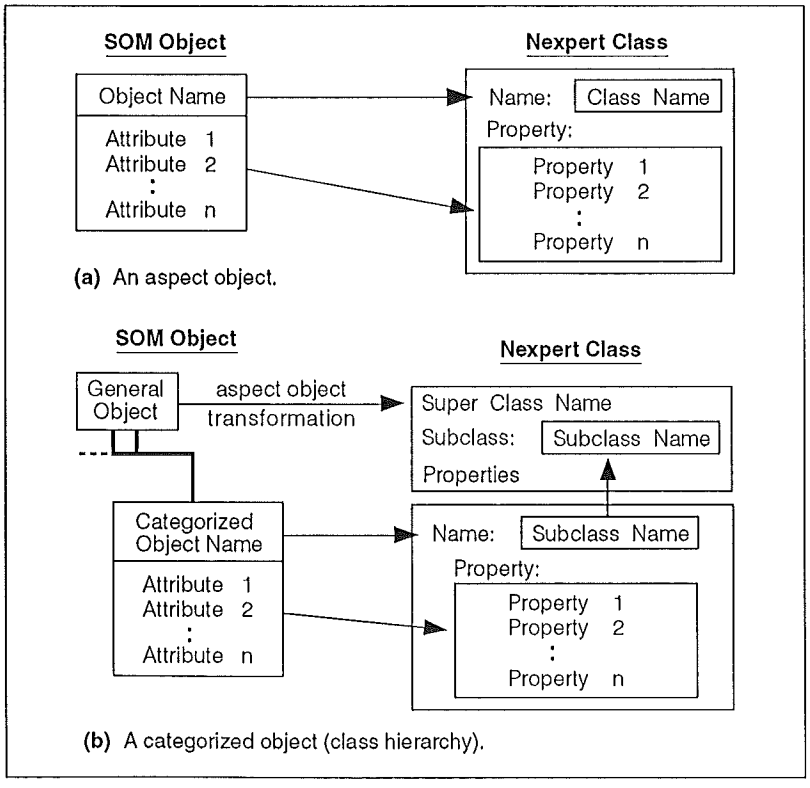


Figure 8. Transformation of SOM Entities to *Nexpert* Classes

strategies based on the SOM diagram and the development environment. *Nexpert* does not support object methods directly, but rather through rules associated with objects. These rules can be structured into a rule-net, a network of rules supporting both forward, backward, or combined forward/backward chaining. Before a rule-net can be developed, general inferencing patterns for the class/object structure must be identified. Suppose two keywords are entered as input data for the prototype implementation. The following four Inference Patterns exist:

1. *Keywords are Strategy type and Research Area type* (Figure 9a). Given a strategy and a research area, multiple project approaches can be identified through the APPROACH-STRATEGY generated. These PIDs would then be used to query the PROJECT-REPORT class.
2. *Keywords are Research Area and Research Tool* (Figure 9b). Given a Research_Area and Research_Tool, the area keyword will be used within the APPROACH-AREA class to first determine corresponding research approaches. The resulting approaches will then be used in conjunction with the tool keyword in the PROJECT-TOOL-APPROACH class to determine the PIDs, which will then determine a list of report IDs in the PROJECT-REPORT class.
3. *Keywords are Strategy and Research Tool* (Figure 9c). Similar to the example shown in Figure 9b, this pattern shows the strategy keyword instantiating values for research approaches; these are then combined with the tool keyword to generate PIDs and then Report-IDs.
4. *Keywords are Research Tool and Research Approach* (Figure 9d). The values for research tool and research approach instantiate many PIDs within PROJECT-TOOL-APPROACH. These PIDs then generate a series of Report-IDs from PROJECT-REPORT.

A rule-net is a realization of inference patterns. The development of a rule-net is a complex task and requires a rigorous analysis and design process. Without the help of a structured design method, a rule-net tends to become a massive "spaghetti" rule-net. Development and maintenance of such a rule-net is very expensive. Although SOM is not yet a complete structured design method for rule-nets, it provides a semi-structured design methodology.

Transformations of various inference patterns to rule-nets are explained as follows:

Single Class. Figure 10 shows the simplest rule-net, which involves a single class. The rule-net box in this figure consists of the left-hand side (LHS), which contains an "IF" clause, and the right-hand side (RHS), which has a hypothesis name and actions clause. When

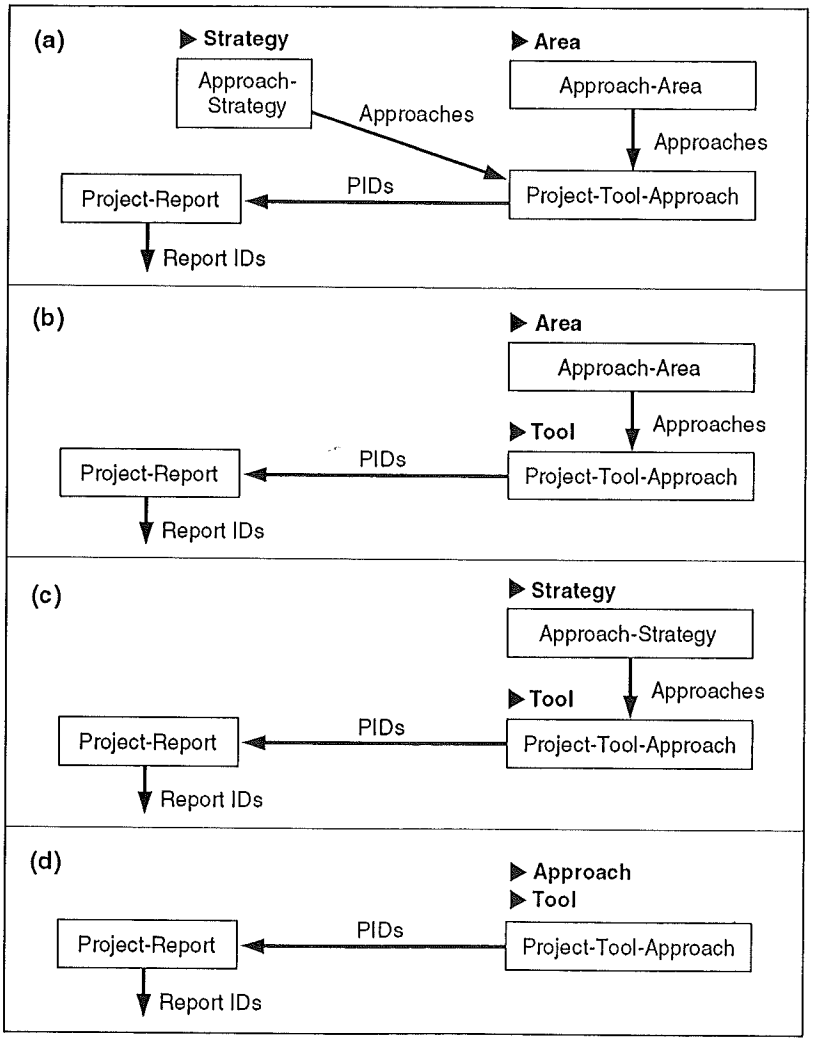


Figure 9. Inference Patterns for Keyword Search

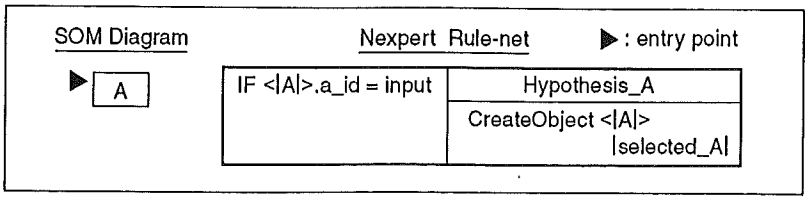


Figure 10. A Single Class Rule-net

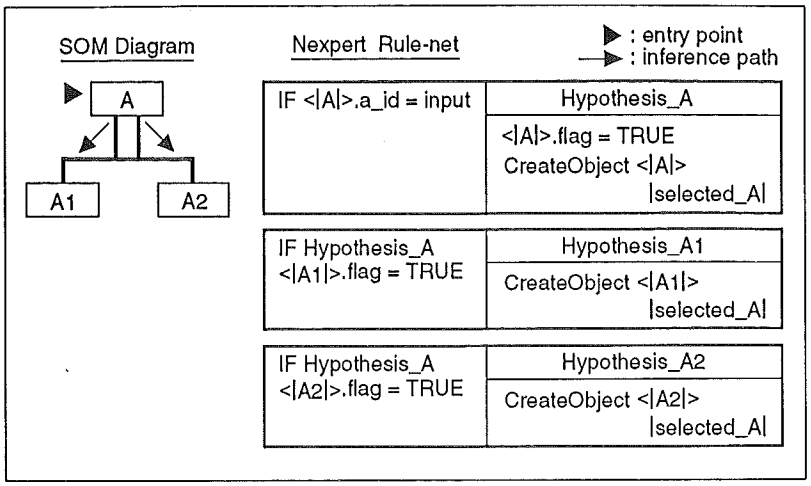


Figure 11. A Subclass Rule-net

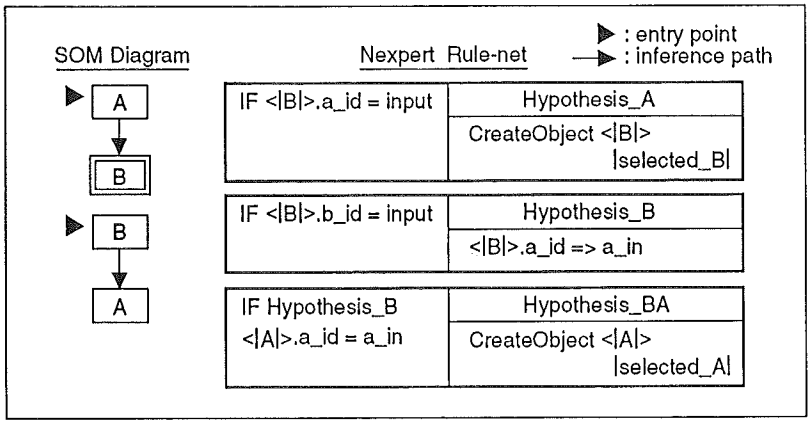


Figure 12. A One-to-Many Rule-net

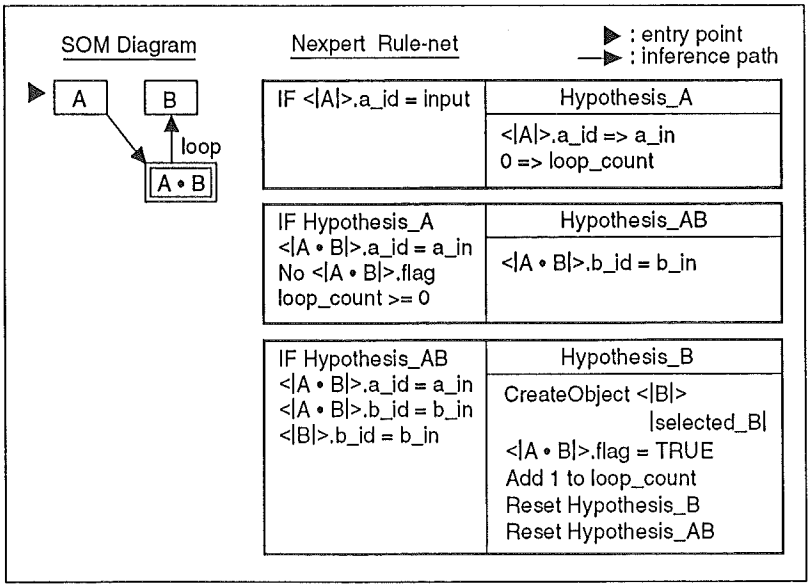


Figure 13. A Many-to-Many Rule-net with One Inference Pattern

LHS is "True," the hypothesis also becomes "True" and actions are executed. In LHS, "input" is an input datum and "a_id" is class A's identifier property. "⟨|A|⟩" represents a list of class A objects. This rule-net instantiates a class-A object whose "a_id" has the same value as the "input." "⟨|A|⟩" in RHS contains only instantiated class A objects that satisfy the LHS condition. "CreateObject" in RHS creates a list of "selected_A" objects using property values from the instantiated class A objects. Definition of the "selected_A" may depend on the actual application; however, by default, it can be defined as a subclass of class A.

Subclass. An inference pattern which involves a class and its subclasses can be represented in the same manner as the rule-net of a single class (see Figure 10). In *Nexpert,* pattern matching in the super class is automatically propagated to its subclasses. However, the collection of instantiated subclasses must be separately performed as shown in Figure 11. In this figure, the "flag" is a Boolean property of class A whose initial value must be set to FALSE at its class property definition. The "selected_A" may contain all properties of class A and its subclasses, although it is not required to (i.e., the selected_A can have only selected properties from the class and subclasses).

One-to-many: The "input" can enter from the "one" side or from the "many" side in this inference pattern (see Figure 12). If it enters from the "one" side (class A), class B objects are directly instantiated using the "input" value because class B already contains "a_id", a foreign key, as its property. If the "input" enters from the "many" side (class B), then the "a_id" is extracted from the class B object and stored in the "a_in." The "a_in" is a *Nexpert* object which must be defined when this rule definition is completed. Since "a_in" is an identifier of class A, a single class A object will be instantiated at the succeeding rule.

Many-to-Many I: An inference

pattern between two classes which have a many-to-many relationship requires a loop. In Figure 13, class A-B is an intersection class resulting from a many-to-many relationship. The loop starts from the intersection class, A-B, and ends at the goal class, B. To make the loop work, increments of a loop counter and resetting of hypotheses must be performed at the loop-end rule, which has the Hypothesis_AB. Otherwise, the loop will be executed only once. The loop terminating condition is determined by the value of the ⟨|A-B|⟩.flag. This condition is checked at the Figure 13 loop-start rule which has the Hypothesis_B. Omission of this process, of course, will cause an infinite loop, so the flags are initialized to FALSE at the class property definition.

Many-to-Many II: Figure 14 shows two inference patterns in a many-to-many situation. Two initial rules, (1) and (i), and two loop-start rules, (2) and (ii), must be prepared. However, the loop-end rule, (3)/(iii), can be used for both inference patterns. In order to use the common loop-end rule, both loop-start rules must have the same hypothesis name (Hypothesis_AB).

Combinations of the five basic Inference Patterns (IPs) discussed in this subsection can be used to perform tasks which require more complex IPs. The SOM methodology is more effective for designing complex domain problems that require a problem-solving strategy.

Methodology Strengths and Weaknesses

The proposed methodology successfully shows a means for representing and applying structural database knowledge. A similar knowledge structure could possibly have been derived using other database representation models, but we believe that the hierarchical SOM structure made a knowledge representation that was much more amenable to the development of searching and inferencing strategies.

Unfortunately, our methodology has only been illustrated for a single (albeit important) user application. Other knowledge representation models and methodologies may prove superior for other applications. Additionally, our implementation examples clearly show that KB implementation strategies are environment-specific. We invite others to investigate the generalizability of this methodology to help determine whether KB/DB coupling methods and implementations are amenable to a general methodology, or if they will remain intractably domain- and application-specific problems.

Related Research

KB/DB coupling using SOM is also being investigated using *Exsys* [14], a nonobject-oriented expert system shell. It appears that a SOM schema can also be translated to nonobject-oriented knowledge bases. Some efficiency is lost when objects cannot be used to reduce the number of rules that must be investigated in each inferencing cycle. Issues that must be resolved include speed of KB development using conventional object-oriented programming environments such as *PC Scheme* versus rapid-prototyping environments such as *Nexpert* and *Exsys;* ease of maintenance for object-oriented knowledge bases versus "flat" knowledge bases; and, performance of systems in each environment.

Other related research efforts are under way. One involves automating coupled KB/DB design through tools such as an icon-based design system for generating a DB schema and knowledge structure using SOM, and an automated SOM structure interpreter that will translate a SOM structure into *Nexpert* object-nets and rule-nets. Another effort is investigating ways of incorporating higher levels of intensional knowledge in the knowledge base, and having this knowledge build upon structural knowledge to provide dynamically defined user queries and back-end processing of retrieved database data. Because some activities are still in progress and others have only been started, definite conclusions cannot be currently drawn.

Conclusions

The implementation of coupled KB/DB systems is a difficult and

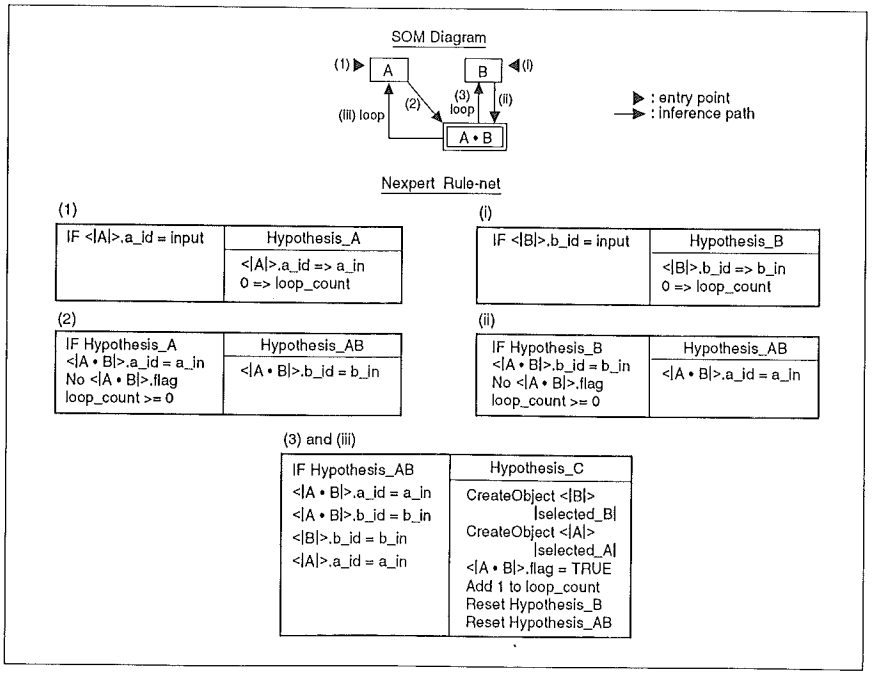


Figure 14. A Many-to-Many Rule-net with Two Inference Patterns

critical task in the development of effective and portable knowledge-based systems. Previous implementations have not suggested a generalizable methodology for developing coupled systems, possibly because traditional database system development environments do not support such implementations. However, recent interest in object-oriented design and implementation environments is revolutionizing the development of coupled KB/DB systems. Object-oriented concepts such as class and object structures, inheritance, and data encapsulation make separation and coupling knowledge and facts smooth and natural. Combining an object-oriented approach with a hierarchical structure further aids the KB/DB coupling process by helping to identify search and inference patterns that can then be encapsulated into object attributes and methods.

This study has suggested a methodology for representing structural database knowledge in a knowledge base and applying this knowledge to an example user application. However, this is only a small part of the overall KB/DB coupling problem. Areas that we have not yet addressed include generic development methods for conventional expert system shells, development methods for a general set of user applications, methods for incorporation of higher levels of intensional knowledge into the knowledge base, and ways to use this methodology with existing conventional-architecture databases. Some of these areas are being addressed in our related research efforts. However, the work we have done in this study can be used to guide the abstract design process of database designers and knowledge engineers attempting to incorporate intelligent system concepts into organizational databases, and possibly motivate the research efforts of others. ◻

References

1. Al-Zobaidie, A. and Grimson, J.B. Expert systems and database systems: How can they serve each other? *Expert Syst. 4,* 1 (Feb. 1987), 30–37.
2. Bic, L. and Gilbert, J.P. Learning from AI: New trends in database technology. *IEEE Comput.* (Mar. 1986), 44–54.
3. Bocca, J., Decker, H., Nicholas, J.M., Vielle, L., and Wallace, M. Some steps towards a DBMS based KBMS. In *Information Processing '86* (Proceedings of the IFIP 10th World Congress, Dublin, Ireland, Sept. 1–5, 1986, Kugler, H.J., Ed.), pp. 1061–1067.
4. Booch, G. *Object Oriented Design.* Benjamin/Cummings Publishing Co., Inc., Redwood City, Calif., 1991.
5. Chen, H. and Dhar, V. A knowledge-based approach to the design of document-based retrieval systems. In *Proceedings of the Fifth Conference on Office Information Systems,* ACM Press, (1990), pp. 281–290.
6. Chen, P. The entity-relationship model-toward a unified view of data. *ACM Trans. Database Sys. 1,* 1 (Mar. 1976), 9–36.
7. Chiaramella, Y. and Defude, B. A prototype of an intelligent system for information retrieval: IOTA. *Info. Proc. Manag. 23,* 4 (1987), 285–303.
8. Codd, E.F. Extending the database relational model to capture more meaning. *ACM Trans. Database Sys. 4,* 4 (Dec. 1979).
9. Coad, P. and Yourdon, E. *Object-Oriented Analysis.* Prentice-Hall, Englewood Cliffs, N.J., 1990.
10. Cohen, B. Merging expert systems and databases. *AI Expert 4,* 2 (Feb. 1989), 22–31.
11. Cohen, P.R. and Kjeldsen, R. Information retrieval by constrained spreading activation in semantic networks. *Info. Proc. Manag. 23,* 4 (1987), 255–268.
12. Croft, W.B. and Thompson, R.H. I^3R: A new approach to the design of document retrieval systems. *J. Amer. Soc. Info. Sci. 38,* 6 (1987), 389–404.
13. Di Benigno, G., Cross, R., and Debessonet, C.G. COREL: A conceptual retrieval system. In *Proceedings of the 1986 ACM SIGIR Conference on Research and Development in Information Retrieval* (Pisa, Italy, Sept. 1986), pp. 144–148.
14. Exsys, Inc., P.O. Box 11247, Albuquerque, NM.
15. Fishman, D.H. The DBMS-Expert System Connection. In *New Directions for Database Systems.* G. Ariav and J. Clifford, Eds., Ablex Publishing Corp., Norwood, N.J., 1986.
16. Gorman, K. and Choobineh, J. The Object-Oriented Entity Relationship Model (OOERM). *J. Manag. Info. Syst. 7,* 3 (1991), 41–65.
17. Hammer, M. and McLeod, D. Database description with SDM: A semantic database model. *ACM Trans. Database Syst. 6,* 3 (Sept. 1981), 351–386.
18. Higa, K. and Liu Sheng, O.R. An intelligent database development: An AIMAIL example. In *Expert System and Advanced Data Processing,* M.L. Emrich, A.R. Sadlowe, and L.F. Arrowood, Eds., North-Holland, 1988.
19. Higa, K. and Liu Sheng, O.R. An object-oriented methodology for end-user logical database design: The structured entity model approach. In *Proceedings of Compsac '89* (Orlando, Fla., Sept. 1989).
20. Higa, K. and Liu Sheng, O.R. An object-oriented methodology for database/knowledge base coupling: An implementation of the structured entity model in *Nexpert* system. *Data Base, 20,* 1 (Spring, 1989), 24–29.
21. IEEE Special Issue on Database and Expert Systems. *IEEE Trans. Database Eng. 6,* 4 (Dec. 1983).
22. Jarke, M. and Vassiliou, Y. Coupling Expert Systems and Database Management Systems. In *Artificial Intelligence Applications for Business,* W. Reitman, Ed., Ablex Publishing Corp., Norwood, N.J., 1984.
23. Leung, K.S. and Wong, M.H. An expert system shell using structured knowledge: An object-oriented approach. *IEEE Comput. 23,* 3 (Mar. 1990), 38–47.
24. Liu Sheng, O.R., Motiwalla, L., Nunamaker, J.F., and Vogel, D. A framework to support managerial activities using office information systems. *J. Manag. Infor. Syst. 6,* 3 (Mar. 1989), 45–67.
25. Liu Sheng, O.R. and Higa, K. An analytical and empirical comparison of end-user logical database design methods. *J. Database Admin. 1,* 2 (1990), 1–15.
26. McCune, B.P., Tong, R.M., Dean, J.S., and Shapiro, D.G. RUBRIC: A system for rule-based information

retrieval. *IEEE Trans. Softw. Eng. 11,* 9 (Sept. 1985), 939–944.

27. McDermott, J. R1: A rule-based configurer of computer systems. *Art. Intell. 19,* 1 (1982), 39–88.
28. McGregor, J.D. and Korson, T. Introduction to the special issue on Object-Oriented Design. *Commun. ACM 33,* 9 (Sept. 1990), 38.
29. Monarch, I. and Carbonell, J. COALSORT: A knowledge-based interface. *IEEE Expert* (Spring 1987), 39–53.
30. Morrison, J. and Morrison, M. PROPHET: A flexible maintainable, and expandable object-oriented knowledge base/database system. University of Arizona Working Paper Series, 1990.
31. Morrison, J., Morrison, M., and Sheng, O.R.L. A hierarchical, object-oriented knowledge base architecture for coupled knowledge base/database systems. University of Arizona Working Paper Series, 1990.
32. Neuron Data Inc., 444 High Street, Palo Alto, CA.
33. Nunamaker, J.F., Liu Sheng, O.R., and Amaravadi, C.S. Integrated office information system (IOIS) report on case study of AIRMICS, Technical report submitted to AIRMICS, Sept. 1988.
34. Patel-Schneider, P.F., Brachman, R.J., and Levesque, H.J. ARGON: Knowledge representation meets information retrieval. In *Proceedings of the First Conference on Artificial Intelligence Applications* (Denver, Colorado, Dec. 1984). IEEE Computer Society Press, Washington, DC, pp. 280–286.
35. Pollitt, S. CANSEARCH: An expert systems approach to document retrieval. *Inf. Proc. Manage. 23,* 2 (Feb. 1987), 119–138.
36. Potter, W.D. KDL-Advisor: A knowledge/data based system written in KDL. In *Proceedings Twenty-First Annual Hawaii International Conference on System Sciences and Knowledge-Based Systems Track* (Jan. 1988), pp. 319–328.
37. Ram, S., Hayne, S., and Carlson, D. Integration of information systems technologies to support consultation in an information center. In *Proceedings of the Ninth International Conference on Information Systems,* 1989, pp. 67–77.
38. Ross, R. *Entity Modeling: Techniques and Application.* Database Research Group, Boston, Mass., 1987, 9.
39. Sheu, P.C. Describing semantic data bases with logic. *J. Syst. Softw. 9* (Sept. 1989), 19–27.
40. *SIGART Newsletter.* Special Issue on Databases, 86, Nov. 1983.
41. Simon, H.A. *The Science of the Artificial, 2nd ed.* MIT Press, Cambridge, Mass., 1981.
42. Smith, P.J., Shute, S.J., Galdes, D., and Chignell, M.H. Knowledge-based search tactics for an intelligent intermediary system. *ACM Trans. Office Info. Syst. 7,* 3 (July 1989), 246–270.
43. Texas Instruments Incorporated, Data Systems Group, M/S 2151, P.O. Box 2909, Austin, TX.
44. Tong, R.M. and Shapiro, D.G. Experimental investigations of uncertainty in a rule-based system for information retrieval. *Int. J. Man-Machine Studies 22* (1985), 265–282.
45. Wiederhold, G. Knowledge and database management. *IEEE Softw.* (Jan. 1984), 63–67.
46. Williams, M.D. What makes RABBIT run? *Int. J. Man-Machine Studies 21* (1984) 333–352.
47. Zeigler, B.P. *Multifacetted Modelling and Discrete Event Stimulation.* Academic Press, London, 1984.
48. Zeigler, B.P. System-theoretic representation of simulation models. *IEEE Trans. 16,* 1 (1985), 19–34.
49. Zeigler, B.P. Knowledge representation from Minsky to Newton and beyond. *Appl. Art. Int. 1,* 1 (1987), Hemisphere Press, 87–107.
50. Zeigler, B.P. Object-oriented modelling and discrete event simulation. Prepared for *Advances In Computers, 32,* M.C. Yovits, Ed., (1990).

CR Categories and Subject Descriptors: H.2.1 [**Database Management**]: Local Design—*Data models, schema and subschema:* H.2.8 [**Database Management**]: Database Applications; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query formulation, retrieval models, search process;* I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Representations (procedure and rule-based)*

General Terms: Design

Additional Keywords and Phrases: Coupled knowledge base/database systems, knowledge base design

About the Authors:

KUNIHIKO HIGA is Assistant Professor of Information Technology Management in the School of Management at Georgia Institute of Technology in Atlanta, Georgia. His primary research interests are in the areas of graph-based design methodologies for databases and knowledge bases, and integration of multiple databases. **Author's Present Address:** Georgia Institute of Technology, School of Management, Information Technology Management, Atlanta, GA, 30332-0520; email: khiga@GTRi01

MIKE MORRISON is a doctoral candidate in MIS at the University of Arizona in Tucson, Arizona. His current research interests include software engineering with an emphasis on distributed, multi-use LAN-based systems, interface design, and artificial intelligence from an applications perspective.

JOLINE MORRISON is a doctoral candidate in MIS at the University of Arizona in Tucson, Arizona. Her primary research interests are in the areas of organizational data management, intelligent database systems, and group collaboration support systems.

OLIVIA R. LIU SHENG is Assistant Professor of Management Information Systems at the University of Arizona in Tucson, Arizona. Her current research interests include analysis and design of distributed, heterogeneous, and multimedia database and knowledge base systems, automation of database and knowledge design, and computer-mediated communication support. **Authors' Present Address:** Department of Management Information Systems, University of Arizona, Tucson, AZ 85721; email: {morrisonm, morrisonj, sheng}@ariz.mis

A Bit-Tree Data Structure for Fast File Processing

DAVID E. FERGUSON

Virtually all modern-day file search techniques use some sort of B-Tree structure. In fact, Douglas Comer in the "The Ubiquitous B-Tree" [3] states: "The B-Tree is, de facto, the standard organization for indexes in a database system." There are many variants of the B-Tree as first described by Bayer and McCreight [1]. This article describes one such variant called a "Bit-Tree." Understanding any of these variants requires understanding of the basic B-Tree model. We will describe B-Trees and a more detailed description can be found in Knuth [6].

A "Bit-Tree" is designed to pack more information into leaf nodes. The denser this information can be compacted, the lower the height of the tree with an attendant reduction in the number of input-output operations required to search the tree. As a by-product, less space is required for the tree.

This article will restrict attention to the most common file operations: "get," "add," "delete," and "get equal or greater" (i.e., get the record with key equal to or greater than, the argument key). If key updating is allowed, it is implemented as an "add" followed by a "delete."

Bit-Tree

History

Prefix compression seems to have been first used in a computer program by Robert L. Patrick at Convair Fort Worth circa 1953 in various application programs on the IBM 704.

IBM implemented a method on the IBM System/38 [5] which used bits to distinguish between keys but also carried along enough bytes of the key so that at any point the key could be reconstructed from left to right. This assured that after a search, such as a Prefix B-Tree search, if the search key was not found, then the position found was the correct place to insert a new entry. This contrasts with the Bit-Tree, in which it is necessary to scan right or left to find out where to insert the new entry. The Bit-Tree, however, requires less storage and consequently lower-level trees.

Distinction bits were first used by Donald Morrison in the Practical Algorithm To Retrieve Information Coded in Alphanumeric (PATRICIA) [7]. PATRICIA used a binary tree structure (sometimes called a "trie," from reTRIEval) when the entire trie could be contained in memory. Since storage was not an issue, these tries used left and right pointers and flags. The term trie is often used when entries relate to the representation of keys rather than the relationship of keys.

B-Trees

This description will depart from that given in [1] in that it will describe a B-Tree that more closely represents B-Tree implementations used in contemporary data processing.

A B-Tree consists of a number of nodes. Each node contains a number of pointers separated by keys. Hence, there is one more pointer in a node than there are keys. The keys are in ascending order. Every node is either a *branch* or a *leaf* node. The pointers in a branch node point to other nodes. The pointers in a leaf node are relative record numbers (RRN) that specify the corresponding ordinal record numbers. One node in the tree is the *root* node. Starting at the root, the number of nodes traversed before encountering a leaf node is called the *height* of the tree. All leaf nodes are at the same distance from the root. The number of entries in a node, called the *order* of the tree, is determined by the node size. If key compression is employed, however, the order of different nodes in a tree may vary.

Searching a B-Tree for a record with a particular key is straightforward. Starting with the root node, search each node encountered from left to right until a key is found that is equal to or greater than the argument key. Then, if the node is not a leaf node, take the pointer preceding that key to find the next node to search. After completing the search of a leaf node, the preceding pointer is the relative record number of the data record sought.

If the root node can reside in memory, it is easy to see that the number of reads necessary to find a record is equal to the height of the tree.

Adding a new record to the file is slightly more complicated. Search the tree as before. Then move the pointer that was found and all the entries to the right of it one position (i.e., one key length plus one pointer length) to the right. Insert the record number and the key of the new record into the vacant position. If the addition of this data causes the total data to exceed the node size, the node is split into two nodes.

To split a node, remove the middle key, called the *split* key. Now write all information to the left of the split key as one node, and all information to the right of the split key as another new node. Then insert a pointer to the new node, and the split key into the parent of the original node. If this causes the parent to become overfull, the parent is then split in the same manner and so on. Finally, if this causes the root to split, create a new node consisting of pointers to the two halves of the previous root node, separated by the last split key. This is how the tree grows in height as the file gets larger. Notice that this technique assures that all leaf nodes are at the same level.

In practice, nodes are split when reading down the tree, due to an improvement suggested by Guibas and Sedgwick [4]. This not only simplifies the program, but more important, simplifies the locking protocol required when more than one user is adding a record in the same vicinity of the tree.

Before discussing the compacting technique a Bit-Tree processor uses for leaf nodes, it should be noted that any conventional technique can be used to represent information in the branch nodes. The recommended way to do this is the method described by Bayer and Unterauer in "Prefix B-Trees" [2]. "Prefix B-Trees," however, is a misnomer, since the compression method used could be used in "non-Tree" searching and is not restricted to tree structures (perhaps the authors were being too modest). Several important points are discussed in [2], and we will now briefly highlight them.

Prefix B-Trees

The first point is that in order to search a file by key, it is not necessary to have a list of keys, but rather to be able to distinguish between keys. Suppose for example, that a search has been narrowed down to two keys, "JOSE FLORES" and "JOSEPHSON FLOWERS", so that now it is necessary to decide which to choose. Typically an index sequential access method (ISAM) file would have the key "JOSEPHSON FLOWERS" and this would be sufficient to make the distinction, that is, if the key being searched for is less than this key, take the first entry, otherwise, take the second entry. Notice, however, that any quantity greater than "JOSE FLORES" and equal to or less than "JOSEPHSON FLOWERS" would suffice equally well to enable the correct entry to be selected. For instance the quantity "JOSEPHINE" would do as the distinguishing key.

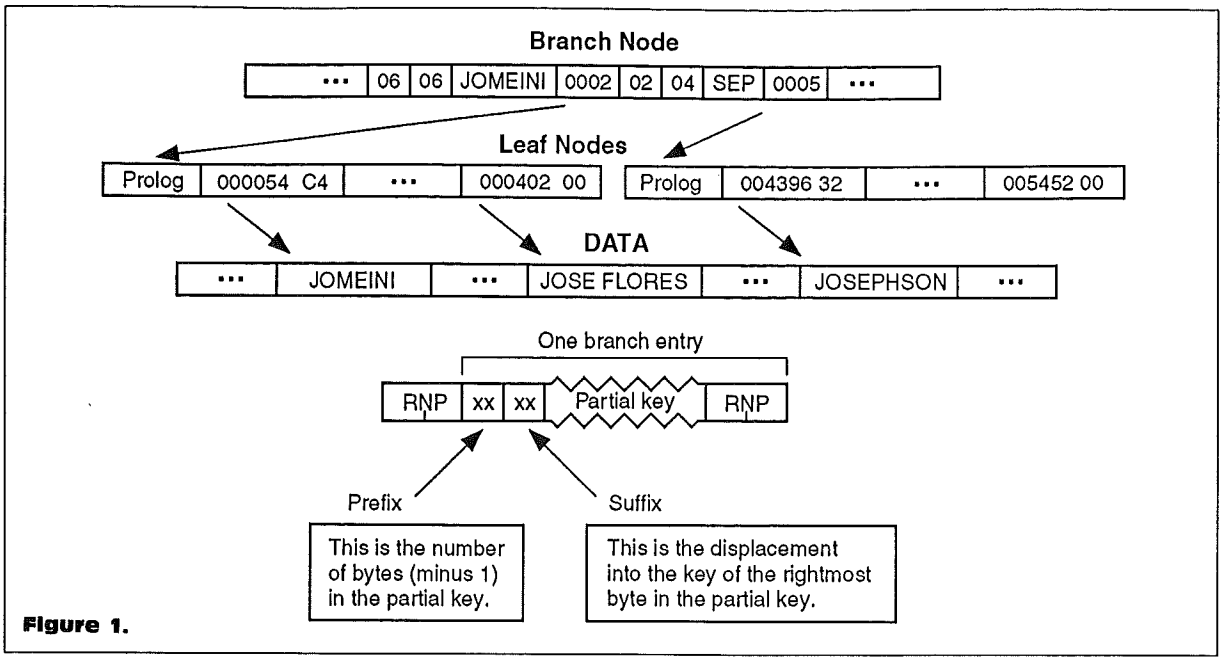


Figure 1.

The second point is that "front end" or "prefix" compression should be used. Suppose "JOSEPHSON FLOWERS" is the first key in a node, and "JOMEINI" is the first key in the prior node. Then the leading two bytes in "JOSEPHSON FLOWERS" are redundant. After prefix compression, the partial key could be "SEPHSON FLOWERS" (or could just as well be "SEPINE").

The third point is that "rear end" or "suffix" compression should be used. It is only necessary to keep enough information in the leading entry in a node to distinguish it from the last entry in the prior node. In the current example, suppose the last entry in the node preceding the node starting with "JOSEPHSON FLOWERS" was "JOSE FLORES". The first byte that differs in these keys is byte 5, or the "P" in "JOSEPHSON FLOWERS" (or in "JOSEPHINE"). Hence, with the prefix compression already determined, the partial key in this case would be "SEP".

This example is shown graphically in Figure 1. Most implementations would keep certain "housekeeping" information such as "node type," "node length," at the beginning of the node. This is shown as "prolog" in Figure 1. In that figure, "RNP" refers to Relative Node Pointer," which is described in the conclusion.

The final point is that when splitting a node, the *split point* can be chosen to maximize the amount of suffix compression. Notice from the previous paragraph that suffix compression is only determined by the last key in the previous node and the first key in the succeeding node. Instead of splitting a node exactly in the middle, suppose a certain range of entries is considered for a possible split point. Then the split point chosen should be between the two keys having the least number of identical leading bytes.

Bit-Trees

Any indexed file search technique uses key information and relative record numbers, (RRNs), to find the desired record. Simply stated, key information is examined in order to find the RRN for the record with the key desired (if it exists). In Bit-Tree leaf nodes, the key information used is the *distinction bit* between two consecutive keys. The distinction bit is defined as the ordinal number of the most significant bit which differs in the two keys, normally with a bias. The purpose of the bias is to reserve distinction bit values of zero so that they can be used to find the beginning and end of nodes. This fact will be used in some of the algorithms described later.

It may seem that Bit-Trees are the logical extension of Prefix B-Trees taken to the conclusion, however, this is not the case. Both Bit-Trees and Prefix B-Trees lose information about the keys. When the search of a Prefix B-Tree node is complete, however, only insignificant information about the last key tested is unknown. That is, the correct position in the current node has been found for any search. Unfortunately, this is not true for Bit-Trees. When the search of a Bit-Tree node is complete, the correct position in the node has only been found if a record with that key is in the file. This of course is disastrous when searching branch nodes for a key that is not in the file (e.g., before making an "add") since one must know which node to search next. It is for this reason that distinction bits are only used in leaf nodes.

Bit-Tree

Any technique that loses information about the keys cannot assure (without reading the data file) that the key searched for is in the file. Using a Bit-Tree or a Prefix B-Tree search, it is at least assured that the RRN found is the correct one when the record is in the file. On the other hand, if no such record is in the file, and it is now to be added, the RRN in the position in the node found (by either of these techniques) is used to read the data record, so that the search key can be compared with the actual key in the data record to determine a "found" or "not found" result.

Assume b (the bias) = 8 and m (the key length) = 8

Key K	1	1	0	1	0	0	1	0
Key P	1	1	0	1	0	1	1	1
Distinction bit						↑		
Bit numbering	0	1	2	3	4	5	6	7
Bit numbering with bias	8	9	10	11	12	13	14	15

$D(K,P) = b + m - 1 - [\log_2(K \oplus P)] = 8 + 8 - 1 - 2 = 13$

Figure 2.

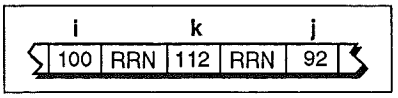


Figure 3.

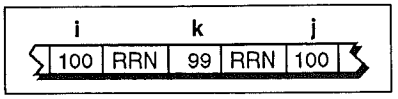


Figure 4.

Bit-Tree Algorithms

Let D(K,L) represent the distinction bit (hereafter called the *D-Bit*) between the keys K and L where K does not equal L. Then if the bits are numbered from left to right starting at 0, and $\oplus$ represents *exclusive or*, D(K,L) can be defined formally as: $D(K,L) = b + m - 1 - [\log_2(K \oplus L)]$ where m is the key length in bits, b is the bias, and [x] means the integer part of x. An example of this is shown in Figure 2 for the (EBCDIC) keys "K" (hex D2) and "P" (hex D7).

If a certain leaf node points to n + 1 records, let R_i, (i = 0, . . . ,n), represent the i'th RRN in the node. Let K_i represent the key of the record specified by R_i. Then, as is usual, the R_is are ordered in the node so that $K_i < K_{i+1}$. Between each pair of RRNs there must be a distinction bit.

Let D_i, (i = 1, . . . ,n), represent $D(K_{i-1},K_i)$. Since the keys are in ascending order and this is the highest-order bit change, it must be from a 0 to a 1, so the D_i bit is always on in K_i. Then the distinction bits $D_i = 100$, $D_k = 112$ and $D_j = 92$ would appear as shown in Figure 3.

Before describing the search technique, it is necessary to make some observations. Suppose that for some i and j with i < j, we have a) $D_j < D_i$ and b) $D_i < D_k$ for all k in the range [i + 1,j − 1]. That is, D_j is the first D-Bit following D_i which is smaller than D_i. The following theorem is required:

THEOREM 1: *For the sequence of D-Bits just mentioned, the D_i bit is on in K_{i+1}, . . . K_{j-1}.*

PROOF: Since D_{i+1} through D_{j-1} are all larger than D_i, none of the lower (more significant) bits could have changed in K_{i+1} through K_{j-1}. Therefore the D_i bit must be on in each of those corresponding keys.

THEOREM 2: *If $D_i = D_j$ there must be a $D_k < D_i$ with $i < k < j$ (see Figure 4).*

PROOF: If there is no smaller distinction bit between D_i and D_j, then from Theorem 1, the D_i bit must be on in K_{i+1} through K_{j-1}. Then, however, the D_i bit would be on in K_{j-1} so that D_j could not equal D_i.

In particular, D_i can never equal D_{i+1}.

Consequently, to search a leaf node for a particular key "K", assign R_0 to the variable R_{found} and then for each D_i starting with i = 1, test the D_i bit in K. If it is on, assign R_i to the variable R_{found} and continue. If it is off, skip the following distinction bits until a smaller one is found. This is because we know from Theorem 1 that the D_i bit is on for each of these entries. When the end of the node is reached, the relative record number in R_{found} is the record number of the record wanted *if the record is in the file.* This is proved by Theorem 3, which is found in appendix A.

Now read the record and compare its key with K. If it is equal, the record is found, otherwise it is not found. If the record is found and the operation is a "get" or "update" (assuming update of keys is not permitted), no further references to the Bit-Tree are required.

If the record was found and the operation was a "delete", then the node must be modified. Suppose record number R_i is to be deleted. Then D_i and D_{i+1} must also be deleted since they refer to the key of R_i which will no longer exist. This must all be replaced with the distinction bit between K_{i-1} and K_{i+1}. No records, however, have to be read due to the following theorem:

THEOREM 4: *When R_i is deleted the new distinction bit between K_{i-1} and K_{i+1} is the smaller of the old D_i and D_{i+1}.*

PROOF: There are two cases according to whether $D_i < D_{i+1}$ or $D_i > D_{i+1}$. (Notice that D_i cannot equal D_{i+1} because of Theorem 2.)

CASE 1: *$D_i < D_{i+1}$*
If $D_i < D_{i+1}$ then the D_i bit is off in K_{i-1} and is on in K_i. Since $D_i < D_{i+1}$, however, D_i is also on in K_{i+1} (by Theorem 1). Furthermore, by definition, no lower number bit differs between K_{i-1} and K_{i+1}. Hence D_i is the distinction bit between K_{i-1} and K_{i+1}.

CASE 2: *$D_i > D_{i+1}$*
If $D_i > D_{i+1}$ then D_{i+1} is on in K_{i+1} and off in K_i. The distinction bit be-

tween K_{i-1} and K_i was D_i, however, which means that all lower bits in K_{i-1} and K_i were the same. Hence D_{i+1} was off in K_{i-1} also. Consequently D_{i+1} is the distinction bit between K_{i-1} and K_{i+1}.

Combining the results of Case 1 and Case 2, if R_i is deleted, the resulting distinction bit between K_{i-1} and K_{i+1} is the smaller of D_i and D_{i+1}.

Suppose the record was not found, and the operation was a "get equal or greater". First compute the distinction bit, D, between the search key, K, and K_i, the key found by the search routine. If $K > K_i$ then clearly K belongs to the right of K_i. Therefore, scan the entries following D_i, skipping those whose distinction bits are greater than D because, like K_i, they too will have the D-bit off (or else their distinction bit would be D) and hence are also smaller than K. The record desired is the first following record with a distinction bit smaller than D. Notice that a distinction bit equal to D cannot occur because then the entry R_i would not have been selected.

Conversely, suppose $K < K_i$. Then clearly K belongs to the left of K_i. Scanning backwards starting with D_i, let D_j be the first distinction bit found that is smaller than D. Then, since the D bit is on in K_i it must be on in all of the keys K_j through K_i because no distinction bits as small as D were found in this interval. Since the D bit is on in K_j and off in K, $K < K_j$. Also, since D_j is off in K_{j-1} and on in K, $K_{j-1} < K$. Therefore R_j is the record number wanted. (This argument holds even if $i = j$.)

It is because of the preceding algorithm that distinction bits should be biased. Since distinction bit values are recorded in an unsigned field, the smallest possible value is zero. Biasing will preclude any distinction bit from being zero. Then the first record number in the node can be preceded by a zero and the last record number followed by a zero. This assures that the above forward and backward searches for a smaller distinction bit will be successful. These two entries could be called D_0 and D_{n+1}.

Following a "not found" condition, an "add" to the file proceeds as in the last case for a "get equal or greater". When the next higher key is found, insert the distinction bit computed above, D, and the new record number at that point.

Node Splitting

The merit of B-Trees is based on the concept of *splitting nodes*. A split strategy was described for Prefix B-Trees. This showed how to find where to split a node in order to minimize the length of the partial key to be inserted into its parent node. A similar technique can be used with Bit-Trees, although it is by no means obvious, because the keys are not visible. It would be too inefficient to read each candidate key from the data file to find the one that generates the shortest partial key for the parent.

Fortunately, this need not be done. The most efficient split key is K_i where i is such that D_i is a minimum in the range of candidate split points. (Notice that D_i is assured to be unique due to Theorem 2.) What is of concern here is suffix compression. The last byte of the split key that needs to be included in the parent is the first byte that differs from the previous key. By picking the smallest D-bit, we assure that the number of bytes needed is minimized, even without knowing the key of R_{i-1}. Specifically the last byte that needs to be included from the split key is byte B_j where $j = [D_i/8]$ for 8-bit bytes biased at 8.

Conclusion

The author has implemented a Bit-Tree for the IBM System/36. IBM imposes the following restrictions on keyed files: key length less than 121 bytes, file size less than 16,000,000 records. Hence an RRN can be contained in 3 bytes. Also, a tree for a 16 million record file will have less than 65,536 nodes so that an RNP can fit into 2 bytes. The Bit-Tree mentioned has the following characteristics: Nodes are 2,048 bytes. A prolog of 16 bytes is used so that 2,032 bytes are available for entries. For keys up to 30 bytes, leaf node entries are 4 bytes (1-byte distinction bit, 3-byte RRN) and 5 bytes for keys from 30 to 120 bytes (2-byte distinction bit). Almost all files encountered have keys of 30 bytes or less.

Prefix B-Tree type branch nodes are used. Empirical studies show that the compressed keys used in branch nodes average 2 bytes, regardless of key length. Then a branch node entry that averages 6 bytes is comprised as follows: 1 byte specifying bytes in partial key, 1 byte specifying where key ends, 2 bytes for partial key (on average) and 2 bytes for an RNP. (Notice if this technique was used for leaf nodes, entries would have to be 1 byte longer since RRNs would have to be used instead of RNPs.) Therefore, a maximum of 2,032/6 or 338 entries can be contained in a branch node. Also, a minimum of 169 entries and an average of 253 entries can be contained in a branch node.

As a result, all files encountered so far fit into three-level trees. With this implementation, the maximum entries per leaf node is 508 while the minimum is 254 so that the average should be 381. A branch (root) node must be full for a two-level tree to split into a three-level tree, and the preceding paragraph shows that this happens when it has 338 entries which would point to 339 leaves. Hence three-level trees would start at about $339 \times 382 = 129{,}498$ entries. Since the root node is always held resident, a search of files of up to 129,498 records normally requires one input operation to determine the desired record, and one more to access it.

If prefix B-tree type nodes were used for the leaves, then an average entry would require 7 bytes, and three-level trees would be reached for files of about $339 \times 291 = 98{,}649$ records. Consequently, for files between about 98,649 and 129,498 records, using B-Tree type nodes would require an additional

disk access.

The preceding paragraph, however, uses the average length of a B-Tree type entry. Whereas the entry length of a Bit-Tree entry is fixed at 4 bytes (for keys up to 30 bytes), the entry length for a B-Tree type entry can approach K/2 where K is the key length. This can be seen by considering a file comprised of pairs of keys differing in the last byte, and one pair differing from the next in one of the first bytes.

Finally, a fair amount of computing time is spent searching leaf nodes. Because Bit-Tree entries are a fixed length, the coding can be much more efficient. In the implementation described here, the number of cycles to inspect a Prefix B-Tree entry is 184 while the number of cycles to inspect a Bit-Tree entry is 42. Furthermore, inspecting Prefix B-Tree entries requires instruction modification on the IBM System/36 whereas inspecting Bit-Tree entries does not. This can be disastrous for reentrant programs.

Using a Bit-Tree, an additional input operation must be done for an "add" operation. (This additional input operation must also be done for a "get". Notice that in this case, this is exactly the same operation that would be done next anyway.) In most applications, however, "get" operations happen far more often than "add" operations.

Appendix A

THEOREM 3: *If the entry for the key Q (Query key) is in the leaf, this leaf search algorithm will find it: i.e.,* R_{found} *will be set to the record with key Q.*

PROOF: Since R_0 is assigned to R_{found} at the start of the search, there will be some key F (Found key) in R_{found}. The proof is by *reductio ad absurdum;* We assume that $Q \neq F$ and show that F could not have been chosen by the algorithm.

CASE 1. *Let $Q < F$. Since the keys are in ascending order, If Q is the i^{th} entry and F is the j^{th} entry, $i < j$. By definition, all bit positions to the left of D(Q,F) are the same in Q and F. This implies that bit positions 0, 1, 2, . . . , D(Q,F) − 1 are identical for all keys in the i^{th} through j^{th} entries. Hence for $i < k \leq j$, we have $D_k \geq D(Q,F)$.*

It is clear that there must exist an entry m, $i < m \leq j$, such that $D_m = D(Q,F)$, since the D(Q,F) bit must change from 0 in Q to 1 in F at some point in the sequence of keys.

Now consider two subcases.

1. Suppose the algorithm looks at D_m (i.e., it checks the m^{th} entry). Then it will find that the $(D_m)^{th}$ (that is, the $D(Q,F)^{th}$) bit is 0 in the key Q for which we are searching. It will then skip all following entries until it finds a distinction bit which is strictly less than D(Q,F). Hence if $m < j$, the algorithm will skip the j^{th} entry. If $m = j$, it will not make that entry R_{found} since the $(D_m)^{th}$ bit is 0 in Q. In either case, F (the j^{th} entry) would not have been assigned to R_{found}, which is contrary to our assumption.

2. Suppose the algorithm does not look at the m^{th} entry. The only reason would be that the algorithm is skipping all entries until it finds some distinction bit which is less than D_m (which is equal to D(Q,F)). Then it would also skip the j^{th} entry, and F would not be chosen, again contrary to our assumption.

CASE 2. *Suppose $F < Q$. Let F be the i^{th} entry and Q be the j^{th} entry, so again $i < j$. Reasoning as in the beginning of Case 1 we see that for $i < k \leq j$, we have $D_k \geq D(Q,F)$, and that there exists an m, with $i < m \leq j$, such that $D_m = D(Q,F)$.*

Again, consider two subcases:

1. Suppose the algorithm looks at the m^{th} entry. Then since the D_m bit is on in Q, the algorithm at that point assigns the m^{th} entry to R_{found}. Thus F could not have been the value chosen.

2. Suppose the algorithm does not look at the m^{th} entry. This can happen only if the last distinction bit looked at, D_k, was less than D_m. All the bits in the 0, 1, 2, . . ., D(Q,F) − 1, however, are the same for the keys in the i^{th} through j^{th} entries, so $k < i$. Thus, the algorithm must also have skipped the i^{th} entry, again contrary to our assumption. ■

References

1. Bayer, R., and McCreight, E. Organization and maintenance of large ordered indexes. *Acta Inform. 1,* 3 (1972), 173–189.
2. Bayer, R., and Unterauer, K. Prefix B-Trees. *ACM Trans. Datab. Syst. 2,* (Mar. 1977), 11–26.
3. Comer, D. The ubiquitous B-Tree. *Comput. Sur. 2,* 2 (June 1979), 122.
4. Guibas, L., and Sedgewick, R. A dichromatic framework for balanced trees. In *Proceedings of the 19th Symp Foundations of Computer Science,* 1978, 8–21.
5. Howard and Borgendale, System/38 machine indexing support, IBM System/38 Technical Developments, 1980, 67–69.
6. Knuth, D. *The art of computer programming. Vol. 3.* Addison Wesley, Mass. 1973, 478.
7. Morrison, D. PATRICIA-Practical Algorithm To Retrieve Information Coded In Alphanumeric, *J. ACM 15* (1968), 514–534.

CR Categories and Subject Descriptors: D.4.3 [**Operating Systems Management**]: File Systems Management—*file organization;* E.5 [**Data**]: Files—*sorting/searching;* H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*search process*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: B-Tree, Bit-Tree, branch, distinction bit, index, information retrieval, ISAM, leaf, mode, random access, root, trie

About the Author

DAVID E. FERGUSON is chairman and CEO at Amalgamated Software of North America, Inc., in Big Bear Lake, Calif. His research concentrates on operating systems, principally sorting, searching, compilers and assemblers, and on number theory. **Author's Present Address:** Amalgamated Software of North America, Inc., P.O. Box 1668, 611 Spruce Road, Big Bear Lake, CA 92315

Debunking the Software Patent Myths

PAUL HECKEL

Jealousy and Envy deny the merit or the novelty of your invention; but Vanity, when the novelty and merit are established, claims it for its own... One would not therefore, of all faculties, or qualities of the mind, wish for a friend, or a child, that he should have that of invention. For his attempts to benefit mankind in that way, however well imagined, if they do not succeed, expose him, though very unjustly, to general ridicule and contempt; and if they do succeed, to envy, robbery, and abuse.
BEN FRANKLIN, 1775 [6]

The issue of software patentability is an important topic because it affects the environment in which programmers and designers work, software innovation, the health of the software industry, and U.S. competitiveness. While the writing of this article was motivated by "Against Software Patents," by the League for Programming Freedom in the Jan. 1992 issue of *Communications* it is an overall defense of software patents.

An Absurd Patent

U.S. Patent 4,736,308, the first patent under the heading "Absurd Patents" in "Against Software Patents," is described: "For example, Apple was sued because the HyperCard program allegedly violates patent number 4,736,308, a patent that covers displaying portions of two or more strings together on the screen, effectively scrolling with multiple subwindows. Scrolling and subwindows are well-known techniques, but combining them is apparently illegal." The League calls this an "outrageous result." Based on this description alone, any reasonable person would have to agree.

But I am that inventor and Apple was actually sued on a prior related patent, 4,486,857. Because my patents were misrepresented, I researched the other patents described in the League's article and am reporting my results.

There is much the League did not say about my patent and the circumstances surrounding it. First, it did not describe my background. In 1963 I worked on the software for the first computer designed to be a timesharing computer. I was at Xerox PARC in its early days, wrote two articles for *Communications* [17,20] and a book on user interface

design [16]. My patent covers a commercial product called Zoomracks [19], which introduced a new computer metaphor called the card and rack metaphor. Zoomracks was marketed primarily on the Atari ST. Zoomracks developed a strong base of users who used it for a very broad range of applications, but it was a financial struggle largely because Atari did poorly. In Aug. 1987, Apple Computer introduced HyperCard, which is based on a similar, but more limited card and stack version of the metaphor.

I was then faced with having invested six years of raising money, developing a product, marketing it, and proving its value in the market, only to find I was in debt, my customer base was on a dying computer and Apple was giving away *free* a more polished and featured, although less elegant, version of the metaphor. While Apple may not have set out to rip off Zoomracks, it was aware of Zoomracks (having seen it under nondisclosure), of HyperCard's similarity to Zoomracks, and that Zoomracks was protected by patents.

HyperCard created expectations that Zoomracks could not meet, and other companies began to develop HyperCard clones. Meanwhile, I asserted my rights, sued and settled with Apple, licensing the patents. Apple is to be applauded for respecting my patents.

IBM was less respectful: We had twice brought our patent to its attention with respect to products like HyperCard and we had visibly asserted our patents and sued and settled with Apple by the time IBM decided to bundle what many consider to be a HyperCard clone. If this article has an anti-IBM patina to it, it is because I spent six months patiently trying to deal with IBM. Finally, IBM representatives flew to San Francisco to show us prior art—earlier technology—invalidating our patents that they claimed to have. When they arrived, they refused to show us the prior art, "for fear the patent office would recertify our patents in error." Even if IBM had been straightforward with me during the six months, to accept such an assertion without evidence would have been naive.

Faced with a choice of accepting IBM's offer of 0.2% of the $5 million IBM is said to have paid to license the token ring patent, or to accept its challenge to "sue us" if we wanted to see the prior art, IBM left me no choice but to fight. But I have chosen to fight in the court of public opinion where possible, rather than the civil courts where, because of its financial strength, IBM has the detailed advantage. I added a description of my dealings with IBM to my book [16] and later sent copies of my book to the members of the Commission on Patent Reform when they asked for comments.

Based on my experience I formulated Heckel's Principle of Dealing with Big Companies: *There is no such thing as a free lunch; unless you're the lunch.*

With Apple and IBM, I did battle against large companies who were sophisticated about intellectual property, rather than small ones that were not. I felt it was in everyone's interest to force companies and the courts to make decisions about software patents so the rules and the marketplace realities can be clear to all, not just the sophisticated few. This article is written in that same spirit. While it is a personal issue, I write to clarify the software patent issues in general, to raise the level of discussion and because like most good inventors, I am curious about what the truth is.

One can only understand the need for patents in light of the competitive marketplace. We need a heavy to show what the innovator faces, just as Humphrey Bogart needed Sidney Greenstreet in *The Maltese Falcon.* IBM has already presented itself in that role; it will reappear as did Sidney Greenstreet.

The Informed Opinion

We will visit the other eight patents mentioned by the League in its article and show that the patents it selected, on examination, disprove its case. But first, we take the broader view.

Should software be patentable like other technologies? The primary issue is a policy one and so we have been influenced by Neustadt and May and their book on governmental decision-making [33]. We ask: What analogies (to software) exist? What are the similarities and differences? What are the assumptions, explicit and hidden? "What is known?" "What is the history of the issues?" What are the interests of the various players?" We will follow the Goldberg rule and ask, not "What is the problem?" but "What is the story?" Most important, we should ask, "How did things turn out in the past?"

History and innovation economics, more than law and computer science, must be the foundation on which to make policy. We have framed 10 points which are, we believe, the consensus of informed opinion on software patents. We hope they help you crystallize your thoughts on patents and enable you to better articulate your differences, if any, with the informed opinion.

1. By creating property rights, patents promote innovation in non-software areas. They particularly promote innovation from small and mid-size companies.

Most of the arguments against software patents turn out to be arguments against patents *per se.* These arguments are advanced most credibly on the basis of established technologies where data and research already exist.

Patents have been accepted around the world as promoting innovation. Many giants of U.S. industry such as G.E., AT&T, Polaroid, Xerox and Hewlett-Packard, started as small companies that used patent protection to protect their inventions.

Yet, most of the articles on patents in the trade, business and even academic press read by the computer community [5, 13, 15, 26, 27,

28, 41, 42, 48] have an antisoftware patent bias. The reason is that for every patent there is one patentholder who is reluctant to speak because the issue is complex and what someone says could be used against him in litigation. And there are a dozen who might like to use the patented technology without paying for it and so are willing to malign the patent and patent system and pass on unsubstantiated rumors and misinformation.

Economists have researched innovation in other technologies [24, 30, 31, 41] and found the following: patents encourage innovation; and small entities—individual inventors and small companies—are a very important source of innovation. According to Jewkes et al. [24],

It is almost impossible to conceive of any existing social institution so faulty in so many ways. It survives only because there seems to be nothing better. And yet for the individual inventor or the small producer struggling to market a new idea, the patent right is crucially important. It is the only resource he possesses and, fragile and precarious as his rights may be, without them he would have nothing by which to establish a claim to a reward for his work. The sale of his ideas directly or the raising of capital for exploiting the ideas would be hopeless without the patent.

While several articles discuss software patents and copyrights [8, 46, 47], few have been written for the software, as opposed to the legal, community [11, 16, 37]. Such studies have only rarely been linked to software [7], and, we are unaware of any empirical studies of the effect of software patents on innovation other than this one.

If we are to reject patents in principle, we should argue that case. If we accept patents as promoting innovation elsewhere but not in software, then we should differentiate software from other technologies.

2. Patents have evolved to address concerns raised by those who suspect software patents.

The courts have developed a patent jurisprudence as a unifying mechanism to support many technologies and foster evolutionary improvement while balancing the rights of patentholders and potential infringers.

Patents have a long history (see sidebar). Most of the concerns about patents raised by the League have been raised long ago in the context of other technologies and addressed in case law and legislation and have stood the test of time. The patent system, like MS/DOS, is not perfect. MS/DOS has a long history of evolutionary improvement: It is a derivative of CP/M, which is a derivative of TOPS-20, which is a derivative of the SDS-940 timesharing system, which evolved from the first timesharing system developed at BBN about 1960. Patent jurisprudence has a similar history of evolution.

Part of the value of patents is they are a proven, public domain *standard* of intellectual property protection having a history of improvement over 500 years, compared to the 30 or 40 years of expe-

A Brief History of Patents

Until recently patents were thought to have originated in England and been used only there prior to America, an error propagated by Jefferson, Lincoln and as late as 1948 by the Supreme Court. Recent scholarship shows their Italian origin and their early use in France, Germany, the Netherlands as well as England. Venice granted 10-year monopolies to inventors of silkmaking devices in the 1200s. These early patents were *ad hoc* grants. In 1474 Venice passed its first patent statute. It recognized patents as a matter of right, rather than royal favor, and provided for fines and the destruction of infringing devices. Galileo was granted a patent [6]. In England, the Queen granted so many monopolies to her friends that citizens protested; so England, in 1624, passed the Statute of Monopolies. This prevented the granting of monopolies, but gave people the right to obtain patents on inventions and imports new to the realm. This statute distinguished between monopolies, which it outlawed as taking from the public what it already had, and patents, which it permitted as giving to the public what it did not yet have.

These patent laws were enacted at the end of the dark ages just before the Renaissance in Italy and the Industrial Revolution in England, suggesting that they *stimulated* innovation.

The Founding Fathers also believed that inventions (and writings) belong to their creators inherently—rather than to the state to be granted at its pleasure. This principle was embodied in the Constitution [1, 6], where Article 1, Section 8 says,

> **The Congress shall have the power to promote the progress of science and the useful arts by securing for a limited time to authors and inventors the exclusive right to their respective writings and discoveries.**

Congress has the power, not to grant rights but to secure inherent rights. This is the principle expressed in the Declaration of Independence that "all men are endowed by their creator with certain unalienable rights." In the Federalist Papers, James Madison, in describing the patent powers observed that, "The public good fully coincides . . . with the claims of individuals.

The creators of the Constitution knew history and understood the ways of men and women, and patents—9 of the 13 colonies granted patents. We should give weight to findings of fact embodied in the Constitution. Two concern patents: Patent rights are inherent rights like freedom of speech; and patents promote innovation.

Obviousness: Polaroid vs. Kodak

As an example of how the patent system has evolved to address the concerns the League raises, consider the Polaroid patent which, they say, describes "differences in the number and order of layers of chemical in a film—differences between the technique Kodak was using and those described by previous expired patents." The League says such differences were obvious. The court held otherwise. Kodak could have avoided infringement by using the order described in the earlier, expired, patent the League refers to. Why would Kodak use the new order described in the later patent rather than the earlier one? Why would Polaroid patent it? Could it have been better? This demonstrates three things to those who must deal with patents. First, an active patent is a territorial warning. Second, technology described in expired patents is in the public domain. Third, patents protect the innovator. Polaroid was the innovator in instant photography. Kodak wanted a share of that market. The major obstacle was Polaroid patents. Kodak tried to get too near the fire. Kodak got burned and paid Polaroid over $900 million.

rience developing operating systems. TopView and OS/2, demonstrate how developing a new operating system and crystallizing a new infrastructure around it are fraught with dangers—known and unknown. An infrastructure has crystallized around MS/DOS. It includes developers and consultants who know it, books explaining its use, and commercial products based on it. Similarly, an infrastructure has crystallized around the patent system. It includes patent lawyers, case law examples of valid and invalid, infringed and not infringed patents, and books and articles explaining patents to both lawyers and nonlawyers.

3. Patents are not perfect.

There are problems with the patent system. Only that which is not real is perfect. The patent community and the Patent and Trademark Office (PTO) are aware of the problems and have been working to address them. A Commission on Patent Reform is considering improvements such as better examination procedures and automatic publication of patent applications after 18 to 24 months.

If lack of perfection were a reason to get rid of something, no one would survive his or her teenage years. Other industries find patents useful in spite of these problems; software will too.

Patents, it is said, inhibit standards. They do not; they inhibit the expropriation of intellectual property without just compensation in violation of the Fifth Amendment. Where patents exist standards are created in two ways:

- Where people want a standard that infringes a patent, the standards body usually negotiates an agreement whereby the patentholder in return for having his or her technology required as part of a standard, agrees to make a standard license and rate available to all.
- Often standards are agreed to which do not infringe any intellectual property. The QWERTY keyboard and the standard automobile controls (Steering wheel, brake and accelerator) demonstrate that patents don't inhibit standards creation. Both public domain standards were developed during the working lifetime of Edison who received 1,100 patents.

4. Software is not inherently different from other technologies in the way innovation or patents work.

Arguments that software is different should be treated critically; you can be sure those same arguments will be used by those who do not believe that the protections of the Bill of Rights extend to areas where computers and software are used.

Fred Brooks, following Aristotle, suggested the distinction between essence and accident [3], and that distinction has guided our analysis in the issues raised by the League and the academics (see sidebar). The question is whether the differences between software are essential or accidental in their encouraging innovation. The League says software is different and so should be protected differently. They present two arguments.

A. ***Programs are complex.***

Why, so they are; but so are airplanes, silicon chips, silicon chip fabrication plants, potato chip plants, oil refineries and many things. But people find the patent system beneficial in these other technologies.

B. ***Software is cheap to develop compared to other technologies because it is a cottage industry.***

Other industries have cottage manufacturers and they deal with patents. Outside of software much invention is a cottage industry; about 5,000 independent inventors belong to the 37 organizations that are members of the National Congress of Independent Inventors. And most cottage industries do not rely on invention.

We should no more optimize an intellectual property system for cottage developers than we should for Fortune 500 companies.

When one talks about marketing and maintaining commercial software products, the costs are much greater than the estimates made by the League. At the other extreme, IBM is reported to have spent 2.5 *billion* dollars to develop OS/2, including applications.

It is expensive to develop software if the task is to design it from scratch and make it a success in the market; it is cheap if the task is to clone something that already exists or is precisely specified.

Indeed, that clone software is so

much cheaper to develop argues for the necessity of patent protection if one wants to stimulate the development of products worth cloning.

Making software nonpatentable or subjecting it to a different form of protection creates practical difficulties, rather like a state seceding from the Union and setting up check points on its border. And if one state secedes, they all can. If each technology has its own *sui generis* (unique) form of protection, we would have to set up boundaries between the different technologies and would need rules for what happens at the boundaries.

This situation occurs in software development. Should programmers be able to define their own conventions or should they conform to the system conventions even where they are not optimal? Do new programmers get to define their own conventions just because they were not involved in the original decision? Aren't programmers expected to abide by the conventions so the code will integrate better and others can maintain it later. Of course, as problems surface, it is foolish to resist all change in conventions just because changes have repercussions. Changes are made, but as part of a deliberative process in which the burden of proof is on those who advocate the changes.

The evolution of the law works the same way: computer law is just another subsystem to be integrated into the fabric of jurisprudence.

The problem of having different conventions in different areas is demonstrated by the Cadtrak patent. It is a hardware, rather than a software, patent. It requires a display device but no software to infringe it. A computer can be designed so it does not infringe, although a simple program running on it can. This demonstrates that simple software programs can infringe almost pure hardware patents and suggests the difficulty of drawing a legal distinction between hardware and software.

Pamela Samuelson laments that "patent lawyers [do not] claim software-related inventions in a straightforward manner" [39]. Patent lawyers are faced with a Catch 22 situation because of the line be-

The Academic Debate: Considered Opinion and Advocacy

The Mainstream View

Donald Chisum has written the standard reference on patent law [9] and is frequently quoted in judicial decisions. His expertise is in integrating patent decisions into a coherent view of the patent law across mechanical, electrical, chemical and other technologies as they are handed down. His reputation rests on the soundness of his analysis in predicting how courts will rule.

Chisum views software as one of many technologies and says software is almost as patentable as anything else and to the degree that it is not, it should be [8]. He has two concerns. First, the current decisions uphold most software as patentable in a way that forces patent lawyers and the technical community to focus on legal technicalities rather than the technical ones. Second, he questions whether lack of patentability will create underinvestment in software innovation as compared to other technologies.

His approach is similar to Judge Schwarzer's in determining how to calibrate the credibility of possible "junk science" in a courtroom. The question is not "Does this make sense in isolation?" but "How does it fit into an organized body of knowledge?" [44]

The Contrarian's View

The League provides no citations for its position in "Against Software Patents." But a similar article by the same authors [15] has four citations: Mitchell Kapor's congressional testimony and articles by lawyers Pamela Samuelson [39, 41] and Brian Kahin [26] let us see how the antisoftware patent position fits in with "an organized body of knowledge."

While patent lawyers conversant with software are in virtual unanimity that software should be patentable, neither of these lawyers is a patent lawyer.

As background we should consider the legal education most lawyers get. First, very few study patent law and thus are not exposed to the fundamental concept that patent rights are property rights. Second, most law students do take antitrust law where they learn that monopolies are illegal. Later, when they discover patents they often mistakenly see monopolies. Third, law students are taught to be advocates. Their job is not to make a considered opinion, but to present their case as best they can; someone else—the judge, Congress, the public—makes decisions.

This author's opinion is that those who took an antipatent point of view did it as a knee jerk intuitive reaction. Away from the mainline of software business and patent law, distrustful of monopolies and inexperienced with intellectual property, these lawyers and software developers found support in one another. This view gained respectability as a contrarian's view and is always welcome in legal journals and at conferences.

Samuelson argues that the basis of software patentability is weak and says the primary issue is a policy one [39]. She presents two arguments

against software patentability and Kahin a third:

A. The industry has flourished and innovated without patents.

A more accurate statement would be: "The industry has flourished and innovated *with little realization in the software community* that patent protection was available."

The growth of the software industry was not due to differences between software and other technologies, but to its synergetic relationship with the computer industry and the new business opportunities created by the computer's rapidly decreasing costs. Until recently software has been seen by computer companies as a loss leader. Software created the demand for computers which was, and still is, the dominant industry: but software developers were to be kept fat, dumb and happy: salaries were high, patents were not mentioned, and there was lots of technology to play with.

Accidental Empires shows that the personal computer software successes were achieved by amateurs who were lucky enough to be in the right place at the right time [12]. The successful early PC software companies (a) marketed innovations *pioneered by others* and (b) aggressively pursued their own intellectual property rights: Microsoft's MS/DOS is a derivative of CP/M; and Lotus' 1-2-3 of VisiCalc. Had Digital Research, VisiCorp and Software Arts asserted intellectual property rights as aggressively as Microsoft and Lotus, they might not have been eclipsed by them.

In the early stage of the industrial life cycle, the first person in his garage who acts on the opportunity starts with the biggest and most established company in the business. When there is no established competition, new companies can compete without patents. As the industry matures it becomes difficult for new companies to enter the market without a sustainable advantage such as patents.

The free enterprise system rewarded entrepreneurs whose personal computer software companies were successful, as it should, but that success should not blind us to its nature—bringing the innovations of others to market. We expect that if patents had been more widely used by the software industry, the true innovators would have received a fairer share of the rewards, thus rewarding innovation as well as business savvy. Had patents been more widespread, the software industry would have been more profitable, it would have grown with less of a boom-and-bust cycle, albeit less rapidly and there would have been a greater diversity of software product categories and features.

To base a software intellectual property system just on the experience of the last 10 years would be like raising teenagers in the expectation that their childhood will be repeated?

B. Many in the software industry say software patents will discourage innovation.

Samuelson says,

> *. . . it is primarily from the widespread concerns about the effects of patents from within the industry and the technical community that she has pursued this study questioning the patent protection for computer program-related inventions. [39]*

Samuelson addresses this perception by conducting a survey to see how widespread the perception is, at least in the related area of user interface copyrights [42], rather than attempting to determine if the perception reflects reality. In reporting the perception she gives it more credence, reinforcing any error, and creating more concern. It would seem to be more constructive to research the effect of patents on innovation and business formation in other industries to see how it might apply to software, and analyze software patents that exist for their effect on innovation and new business formation as we do.

It makes as much sense to devise theories of innovation by polling the software community as it does to devise laws of physics by polling people who walk. The intuitive answer is not necessarily the correct one. Ask people, "Assume you are walking at a steady pace holding a ball, and you drop the ball. Will the ball hit the ground in front of you, in back of you, or next to you?" most people will say "in back of me" [32]. A physicist will tell you that Newton's laws predict, "next to me," and can perform an experiment to prove it.

In contrast to Chisum and the mainstream of patent law where software is viewed from a broad perspective of many technologies, Samuelson views the law from inside the software community looking out. She has always advocated narrow protection—arguing as late as 1984 that CONTU's recommendation that software in machine readable form be copyrightable was ill considered [40].

Samuelson acknowledges that the conditions that promoted software innovation until now may be different from those that will promote it in the future.

Samuelson proposes to design a special *(sui generis)* form of protection for software, saying, "It is possible to

design a law that is appropriate to the kind of subject matter that software is." How can one be certain of one's ability to build a skyscraper, if one is uncertain if one is building on bedrock or marshland?

Writing laws is like designing systems. They will have desired and undesired, intended and unintended consequences. The current law, especially *Benson* forces patent lawyers into a Catch 22 dilemma: Write a straightforward patent, and get it rejected as a mathematical algorithm; write one that is patentable subject matter and it will not be straightforward. Attempting to make only software unpatentable will no more prevent practical software patents from issuing and being enforced than prohibition will eliminate alcoholism. Samuelson laments that "patent lawyers [do not] claim software-related inventions in a straightforward manner [39]; but this is like blaming a program's users or the market for not behaving as expected—a sure sign of an inexperienced designer. Law, software, and marketing strategies must be designed the same way the Constitution was: based not on how we would like people to behave, but on how self interested people will actually behave.

Academics such as Samuelson, who pontificate on software patents and want to create a *sui generis* system of protection, seem ready to reinvent the wheel before they understand how a wagon works or the infrastructure of the highway system. Most show no knowledge or understanding of the processes of innovation over hundreds of years in a variety of technologies. Most have little, if any, experience prosecuting (filing), analyzing, or litigating patents. They argue that software is different from other technologies, not from the perspective of the history of innovation and patents over a range of technologies, but from the myopic view of the development of a single technology, largely in a single software marketplace in an atypical decade—the personal computer market in the 1980s.

Ben Franklin described a professor who was so learned he knew the word for horse in five languages. *equus* in Latin, *caballo* in Spanish, *cheval* in French, *cavallo* in Italian, and *Pferd,* in German. He then went out to purchase one, and returned with a cow. Given your experience observing complex software systems develop and how the hype manifests itself in reality, when you hear *sui generis* protection systems being proposed as transport into the twenty-first century, ask yourself: "Will I ride horse or a cow?"

Taking a contrary position and fighting for it, as Samuelson does, is in the highest tradition of the law. In part it is because truth emerges from the debate and if there is no one to debate with, a poor sort of truth will emerge. Yesterday's contrarian view may emerge as the mainstream view. It is like having advocates for a programming language like Forth. It forces identification and discussion of issues and influences the industry—PostScript is Forth-like. But software developers seriously consider programing in Forth only in unusual circumstances.

C. Software helps disseminate information

Brian Kahin, a research fellow in the Science, Technology and Public Policy Program at Harvard University's Kennedy School of Government offers a different reason software should not be patentable [25]:

> *A deeper, more disturbing problem in patenting programs was barely evident before computers became ubiquitous personal tools . . . the computer has developed into a medium for human expression and a mediator of human experience. Thus, what is increasingly at stake in software patents is the generation and flow of information.*

The "barely evident" "problem" was addressed by Lincoln who, in his *Lecture on Discoveries* [23], said:

> *certain inventions and discoveries occurred, of particular value on account of their efficiency in facilitating all other inventions and discoveries. Of these were the arts of writing and of printing—the discovery of America, and the introduction of Patent-laws*

Lincoln not only believed that the patent laws encouraged innovation, but he anticipated Kahin in realizing that inventions which promote dissemination of information are particularly important. Since Lincoln's speech, patents seem to have encouraged many inventions which engender the "generation and flow of information": the telephone (Bell), phonograph (Edison), movie camera (Edison), Xerography (Carlson), radio (Armstrong and Marconi), Phototypesetting (Scheffer), and TV (Philo T. Farnsworth).

> *Even the birth of patenting, Aldus Manutius, the famous Venetian scholar printer for whom the desktop publishing company is named, received two patents—one on a form of Greek type—and a ten year monopoly to use the italic font which he invented.*

Having discovered that he is treading in the footsteps of Lincoln, albeit in the opposite direction, Kahin might, in planning his future travel, consult the work of his colleagues at the Kennedy School of Government, which has guided our analysis [33].

tween what is patentable and what is not: Write a straightforward patent, and get it rejected as pure software because of *Benson;* write one that is patentable subject matter and it will not be straightforward. Attempting to make software unpatentable will no more prevent practical software patents from issuing and being enforced than prohibition will eliminate alcoholism.

The PTO and those patent lawyers who prosecute software patents have much more experience in the nitty-gritty of protecting software than academics. And the PTO and the courts have more experience weaving new technologies into the fabric of the patent system than the software community has in creating forms of intellectual property protection.

The debate in the academic world is described in a sidebar. The mainstream view taken by the practicing and academic patent bar and most computer lawyers on one side and the contrarians led by Samuelson, argue against software having the same breadth of protection as other technologies. The League for Programming Freedom has launched an offense in the debate. Like the Battle of the Bulge, it might appear formidable when seen up close. But while it must be treated seriously, it is the last gasp of a dying cause.

The pioneers in each new technology see that technology as new, different, and central, and expect the world to accommodate it. To some extent the world does. But each new technology slowly becomes woven into the tapestry of knowledge encompassing other technologies—each distinctive in its picture—but using the same threads and the same weave.

5. A nonprofit Marxist economic system is not optimal in promoting innovation in software.

This is the paradox one must confront if one argues software patents decrease innovation. The essential difference between Marxism and capitalism is property rights. Patents and copyrights create intellectual property rights that can be brought, sold, rented and licensed like other property rights. Marxism may be better than capitalism in some areas—certainly not in Russia, but capitalism, with all its flaws, has outperformed Marxism.

The paradox of Marxism is not just a theoretical issue. Stallman, the founder of the *League for Programming Freedom,* heads the Free Software Foundation which is developing and planning to distribute a clone of the Unix™ operating system. AT&T has invested in Unix based on its ownership as manifest in patents and copyrights. AT&T cannot be pleased when Stallman gives away free copies of a clone of a product it invested millions in developing and marketing.

If AT&T had not used patents and user interface copyrights to protect its intellectual property rights, Stallman would have no trouble making and distributing a Unix clone. But AT&T must pay its bills with money it receives from customers and has asserted its rights. If it is acceptable to clone Unix or any program, will anyone invest in new ideas? Should we optimize an intellectual property jurisprudence for, not large entities, not small entities, but companies that distribute free clones of other people's software?

For all his talk about wanting to promote innovation, Stallman seems to get his ideas for technology from AT&T, 1969 and his ideas for intellectual property protection from IBM, 1965.

Many software developers do their work for the fun of it. But the distinction is based, not on the technology, but on amateurism: amateurs flourish in the early stages of a new technology. Professionals have accepted that others work for free, but they bristle if they are expected to work at the same rates.

6. Software, like every technology, has unique problems.

Software patents have unique problems: prior art libraries are limited, the search classification system was designed for hardware patents, few computer scientists are examiners. Still when it gets to specific cases, computer scientists and the PTO see invention similarly. (See Document Comparison)

For the last two years the PTO has been improving the situation. It is improving its prior art search facilities in software, has published a new software classification system, and is actively recruiting computer scientists.

The PTO has still not been able to rid itself of the prejudice against software patents, as patent practitioners in the software area will tell you. It still is conservative in its interpretation of what constitutes patentable subject matter, and has rejected several applications that are being appealed.

7. Legally, software is patentable and it will remain so.

Prior to 1982, about 30 different software-related patent cases went through the Appellate Courts. The range of technologies—seismic, medical, petrochemical, telecommunications, firmware, and software—demonstrate that software is both well grounded in patent law, and basic to the advancement of American Industry. Software has become pervasive in industry, which has been basing business decisions on software's being patentable for 10 to 20 years. This has created a sophisticated broad-based constituency for keeping software patentable. Congress has not given in to demands to make less pervasive technologies, such as biotechnology, unpatentable; it is less likely to do so with software. Software has been clearly patentable longer than it has been copyrightable (See Patents and Copyrights).

Chisum, the leading authority on patents, wrote an article on the patentability of software and concluded:

The continuing confusion over the patentability of computer programming ideas can be laid on the doorsteps of a single Supreme Court decision, Gottschalk vs. Benson, *which held that mathematical algorithms cannot be patented, no matter how new and useful. A*

careful analysis of that decision shows the holding is not supported by any of the authorities on which it relied, that the Court misunderstood the nature of the subject matter before it, and that the Court failed to offer any viable policy justification for excluding by judicial fiat mathematical algorithms from the patent system. The Benson *decision is inconsistent with the later Supreme Court ruling in* Diamond vs. Chakabarty *that the patent system applies impartially to new technologies and that any policy issues for excluding new technologies should be addressed to Congress. Policy considerations indicate that patent protection is appropriate for mathematical algorithms that are useful for computer programming as for other technological innovations* [8].

Chisum is in the mainstream in saying that the courts made a mistake by making software unpatentable. But courts are reluctant to overturn previous decisions directly, and then only after their scope has been eroded. A similar situation where prejudice had been become part of the jurisprudence occurred earlier: *Plessy vs. Furgesson* [1896] held that "separate but equal" facilities for whites and blacks were lawful. The courts did not directly overturn it, but eroded its vitality on a case-by-case basis over a period of years in a number of decisions starting with *Murrey vs. Maryland* while appearing to show respect for *Plessy.* Finally, when faced with *Brown vs. Board of Education* [1954], ample precedent had been created for the Supreme Court to overrule *Plessy* directly.

Samuelson, having asked Chisum to write his article, now attempted to refute him, but after arguing for over 100 pages that the basis for software patentability is weak, was forced to conclude that:

. . . the only principle which seems to have guided the court's decisions is one of upholding the patentability of as many program-related inventions as possible while appearing to show respect for the Supreme Court's decisions. [39]

Samuelson's observation seems to be compelling evidence that while she has not been persuaded that *Benson,* like *Plessy,* is a fundamental error in giving prejudice the force of law, the Court has and will, in due course, reverse it and the original intent of Congress will again become law and statutory. Subject matter will "include anything under the sun that is made by man."

8. Whether or not one agrees that software patents are beneficial, patents are here to stay so we should plan to work with them.

The software community will be best served by articles about how to avoid infringement, how to deal with infringement notices, how to find prior art, how to use patents to protect new ideas, how to differentiate products, and how to make the patent system work better for software (based on experience rather than speculation). In brief, we should direct our energies toward making the system work in order to increase innovation and U.S. competitiveness, rather than fighting patents.

9. The practical effect of continuing to spread misinformation on software patents will be to hurt small developers and U.S. competitiveness in software.

Patents, like a cat's claws, function as weapons when necessary. A declawed cat will not survive in the wild; neither can a defenseless startup once it succeeds and attracts substantial competitors. Patents are not the only defense, but they are vital to innovative startups that must survive. In business, as in the jungle, respect is given only to those who can protect themselves.

Microsoft, IBM and others are applying for patents in quantity. Those who do not understand the situation are not. Many are happy to have software patents attacked. Why let your competitor in on a good thing? Why pioneer new

Document Comparison: An Obvious Patent?

A criticism of the patent system is that computer scientists are qualified to judge invention in software while the PTO is not. In his article [26] Kahin says, the PTO is,

awarding patents merely for automating familiar processes such as . . . comparing documents (Patent No. 4,807,182). But software developers have been routinely automating such [functions] for years.

In fact, the ACM published a refereed paper describing that (hashcoding) technique for comparing two text files albeit for source code, rather than document comparison [17]. It seems that the ACM and the PTO have similar standards of inventiveness.

It so happens I wrote that paper, and I brought it to the attention of the patentholder and (indirectly) to WordPerfect about 10 months before Kahin's article was published. Four companies put on notice about the patent brought the paper to the attention of the patentholder. This suggests that where prior art exists which narrows a patent's scope, it is likely to surface.

Advanced Software, was founded to develop and market DocuComp which uses the patented technology. Its inventor, Cary Queen, a Ph.D. in mathematics has filed over a dozen patents in genetic engineering where he is principle in a startup, Protein Design Labs. He also used the patented hashcoding technique to compare genes to identify similarities.

Cary Queen reports there is more prejudice against patents in software than in biotechnology, where hundreds of startups have been financed, as biotechnology patents are better respected.

product ideas when it is less risky to copy competing products and incorporate useful features once market success is proven.

From the perspective of large companies, a loud voice, such as the League, yelling against software patents can be useful as a means to destroy one's competition.

The Japanese are aggressively filing for U.S. patents on software. While our strength is innovation, Japan's is in adapting innovations and steady improvement. But if they have the improvement patents and we did not file for the basic patents, we lose. If we arrogantly dismiss the Japanese as incapable of creating good software or cavalierly dismiss patents as undesirable, then 20 years from now we will be trying to get back the software market from Japan just as today we are trying to get back the automobile and semiconductor markets. We are not even trying to get back the consumer electronics market.

Who is responsible for the misperception about the desirability and legality of software patents? In a certain sense, it is the League for Programming Freedom. But it knows not what it does. And its arguments are the ghosts of arguments for IBM's corporate self-interest of a bygone era. Is not the origin of the problem IBM's attempt in the 1960s to declaw a competing technology by depriving its practitioners of their constitutional rights as inventors? (See Software Patents: IBM's Role in History.)

If it were just a question of IBM outfoxing its competitors, we might learn our lesson and let it pass. But we think it useful to ask some questions: Is it in the interest of the United States to have strong, competitive, innovative software industry? Is it in IBM's interest? Did IBM use its position on the 1966 Patent Commission to put its corporate self-interest ahead of that of the U.S.? Should IBM be held responsible for its role in creating the current software patent mess? Some have proposed making software patents unenforceable. Might a law making IBM patents unenforceable make more sense? Or a law that would prevent IBM from obtaining patents for a period of time, say 5 or 10 years? At a time when competitiveness with Japan is a major concern, what kind of a message should we send about what happens to those who use their positions on government commissions to sacrifice their country's interest to their corporate self-interest?

Similarly, should we eliminate patents to avoid patent litigation as the League suggests; should we not eliminate all laws so as to avoid all litigation?

10. In considering the issues, we should deal with examples of real patents and, where possible, real infringement where facts for both sides are fairly stated.

If we are to have a meaningful debate on whether software should be patentable, I suggest we take our standards, both of debate and of where the burden of proof lies from Abraham Lincoln:

I do not mean to say we are bound to follow implicitly in whatever our fathers did. To do so would be to discard all the lights of current experience—to reject all progress—all improvement . . . if we would supplant the opinions and policy of our fathers in any case, we should do so upon evidence so conclusive, and argument so clear, that even their great authority, fairly considered and weighed, cannot stand . . .

If any man [believes something], he is right to say so, and to enforce his position by all truthful evidence and fair argument which he can. But he has no right to mislead others, who have less

Software Patents: IBM's Role in History

In the late 1960s when IBM's internal policy was that software should not be patentable, IBM vice president, J. W. Birkenstock, chaired a presidential commission on the patent system which recommended that software should not be patentable. We expect that the other commission members deferred to IBM's expertise on software, just as members of a commission designing an aviary would defer to its most knowledgeable member on birds: the cat.

Congress rejected this view, but three paragraphs of the commission's recommendations (e.g., IBM's corporate policy) found their way into *Gottschalk vs. Benson*, the Supreme Court Decision that limited the patentability of software. At this time IBM had 70% of the computer market, so it is not surprising that CBEMA, the Computer Business Equipment Manufacturers Association filed an *amicus curie* brief against software patents in *Benson*.

From this historical perspective we can see that the conventional wisdom that "software has not been patentable," should be more accurately stated as "it was not in the interest of IBM or other computer manufacturers for people to think software is patentable." We have never seen it pointed out in the debate on software patents that the idea that software is not patentable subject matter was formed in the crucible of IBM's self-interest and corporate policies of an earlier time.

IBM and CBEMA have now rejected the Stallman-primal IBM view [7]. But the damage has been done. The PTO and the industry have not taken software patents seriously until recently, which explains the problems the PTO has had in examining patents and the prejudice against software inventors who assert their patent rights. Many in the software community have been suckered into believing software should not be patentable, while IBM has aggressively but quietly been getting software patents and become the company with the largest software sales.

access to history, and less leisure to study it, into [a] false belief . . . thus substituting falsehood and deception for truthful evidence and fair argument.

What I find most frustrating in this debate is that the mode of argument used against software patents by so many [15, 26, 27, 28] is to throw as much mud against the wall as possible and hope some of it will stick. I have expended some effort here removing some of the mud. I do not claim to have removed it all, but I hope I have wiped away enough to show you the rest will wash off too.

A Study of Nine Software Patents

In its article the League lists nine patents—mine and eight others to make its case. It is unlikely that the members of the League considered the positive side of any of the patents they cited. It is as if they went searching for quarters with heads showing, and finding several, reported their findings without turning any of them over. Here we turn over the other eight quarters in an attempt to produce some empirical results.

U.S. Patent 4,197,590: The inventor founded a company to develop and market what appears to be the first personal computer to write directly from memory to the display. This invention has been widely licensed to the personal computer industry by Cadtrak. The "XOR" is only part of the invention. Cadtrak filed and has won at least one lawsuit against a larger company. The idea behind the "XOR" claims of this patent is simple. A program XORs a cursor icon onto a display device; later a second XOR to the same place erases the cursor, restoring the original display. To move the cursor one XORs the cursor to its old location, then XORs it onto the new location. There are many ways to get around this patent. One can use an underline as a cursor or "logically or" the cursor onto the display, erasing later by rewriting the display with its original information. This approach is fast, lets you change cursor icons easily, accesses the minimum possible data, and requires no space be reserved on the screen for the cursor.

The League says this patent can be infringed in "a few lines of a program." It can be, *but not on a computer that was commercially available at the time the invention was made.* The invention is largely the invention of the frame buffer. As such, it requires hardware which has since become common, making it possible to infringe the XOR claims with a few lines of code. Many, if not most, computer manufacturers including Apple and IBM have taken out licenses which cover programs running on their computers.

This patent illustrates that it is usually easy to design around a patent one accidentally infringes. If this patent, a hardware patent, is a "bad" patent as some claim, it only demonstrates that the electronics industry tolerates "bad" patents because it finds patents beneficial on balance. Software should be able to tolerate "bad" patents similarly. To discard the patent system because some bad patents exist would be the same as suppressing free speech to stamp out lies.

U.S. Patent 4,398,249: This is what the League has mischaracterized as fact, in 'Refrac recalculation patent'. In 1970 Rene Pardo and Remy Landau invented the concept of an array of formulas that would enable businesspeople to write their own programs to create business applications. Although the word was not used, their invention is in essence the modern computer spreadsheet. The fact that the claims cover recalculation is an artifact of how the patent claims were written. Pardo and Landau marketed a commercial spreadsheet-like product based on this technology. This invention has been widely adapted in the personal computer industry—over 250 spreadsheets have been marketed.

This patent was originally rejected by the PTO as a mathematical algorithm and thus unpatentable subject matter. Pardo and Landau felt so strongly about their inventive contribution that they appealed their case *pro se,* which means they, not a lawyer, wrote the brief and argued it before the appeals court. The decision, *In re Pardo* [23] is a major legal precedent which establishes that an invention is patentable whether or not the invention involves software "novelty." If their experience was typical, they were stonewalled when they tried to enforce their patents. This would explain why they approached Refac—a white knight in the fight against the patent pirates. If Pardo and Landau have the same deal Refac offered others, then they can expect to collect royalties only as Refac does.

U.S. Patent 4,633,416: This patent

Patents & Copyrights

Kahin says, "Never before has an industry in which copyright was widely established suddenly been subjected to patenting" [25].

In fact, patent protection for software was established before copyright protection. *Diamond vs. Diehr* (1981) preceded the two most important software copyright cases, *Apple Computer vs. Franklin Computer Company* (1982) and *SAS Institute Inc. vs. S&H Computer Systems Inc* (1985). Even today the case law on user interface copyrights is sparse. Software, or "computer-related" patents were obtained in the 1960s. Martin Goetz of Applied Data Research received U.S. patents 3,380,029 in 1968 on a Sorting System, and 3,533,086 in 1970 on AutoFlow, an automatic flow charting program. (When IBM started giving Roscoe, a flowcharting program, away *free,* Applied Data Research sued for antitrust and, in 1969, settled for about $2 million in damages.) The recalculation patent filed in 1970 was granted in 1983.

is held by Quantel, a company that developed a line of commercial video editing products protected by its patents. Quantel filed for patents when it was small, and it has grown from being a small to a large company because it has used its patents to prevent competitors from using its technology. The first company Quantel sued was much larger than it was.

U.S. Patent 4,777,596: The League tells us XyQuest was notified that its product XyWrite infringed Productivity Software patent, protecting the ability to accept an abbreviation or correct a spelling error by hitting a space bar. When licensing negotiations failed, XyQuest removed the feature from future releases.

Productivity Software was founded in 1984 to develop data input systems where minimal keystroke data input is important. Based on its patented technology, Productivity Software has grown to seven employees and markets 31 specialty products. It has found niches in the medical and legal transcription and the handicapped marketplaces.

Patent problems are generally minor compared to the other problems pioneering companies face. At about the same time XyQuest had postponed introduction of its latest

How Patents Work

Exclusive or territorial rights bestow on their owners a long-term outlook, and create a simple test for determining whether or not to fight. This leads to stable solutions and minimizes inefficient disputes. Such rights occur in many areas. A miner stakes a claim; salespeople have exclusive territories, and professors specialize in areas and are given tenure. Such territorial rights stimulate diversity by encouraging competitors to stake their territorial claims at a distance. Lee De Forest, for example, invented the triode, the amplifying vacuum tube, to avoid infringing Fessenden's spade detector patent [29].

Many mistakenly believe that the patent system protects only "flash of genius" insights. That is not true. In 1952, Congress overrode the "flash of genius" doctrine. Patents are designed, not to stimulate invention directly, but to stimulate their commercialization by giving exclusive rights for 17 years to anyone who invents something new and not obvious. Just as an author need meet a standard of creativity to get a copyright on an original work, a patentholder need meet a standard of nonobviousness to get a patent on something new.

The following discussion of patents is useful in providing an overall understanding of how patent infringement is determined, especially where an overly broad (e.g., "bad") patent may be involved.

Before the PTO will allow your patent application, it does a search (rather like a title search when you buy a house) to find prior art on your invention—what others have done earlier that is disclosed in publications or products. The PTO examines the prior art it finds along with any you send it. If what you did is sufficiently different, it issues claims that delimit the territory of your invention.

The process is rather like finding new territory. Suppose you suddenly landed in Left Fork, North Dakota and found that no one lived there and wanted to claim it as yours. You might try to claim all the land west of the Mississippi. The PTO will likely find that people have lived in nearby states and may issue you a claim to say, North Dakota. Of course, the PTO could allow your claim in error. The too broad claim—all the land west of the Mississippi—will look impressive and could be useful as a source of cash from people impressed by surface rather than substance.

In practice, if you try to enforce the patent against, say, Californians who just discovered gold, they would show the court that people lived in California before you landed in Left Fork. The court will declare your broad claim invalid. Practically speaking, you would not go to court once you realized that people lived in California earlier. You might go back to the PTO to get the patent reissued, showing them the prior art and claiming a smaller territory. The PTO might only allow narrower claims that cover eastern North Dakota, or maybe only Left Fork, North Dakota.

A patent does not necessarily give you rights to what it says it does. Undiscovered prior art might considerably narrow its scope. The advantages of being issued a too broad patent are (a) potential infringers might keep a greater distance than they have to, and (b) you can wait to define the limits of your territory until you know the terrain better. The disadvantage is that you might make business decisions based on your belief that you had rights you did not possess.

There are two ways you can respect a patent: you can avoid infringement or you can take out a license. If you are infringing, the patentholder will usually forgive past infringement if you agree to remove infringing capability. Your show of respect for the patent gives its holder credibility with other infringers.

As a possible infringer you have several courses of action when you face an overly broad (or "bad") patent, or indeed any patent:

1. Ignore the patent problem until confronted with it.

version for about a year so it could upgrade to IBM standards as part of an agreement in which IBM would market XyWrite exclusively. At the last minute IBM reneged on the deal [25].

While the first five patents are held by small entities, the last four patents are held by large entities and they also protected commercial products:

U.S. Patent 4,558,302: UniSys licenses this, the LZW compression patent, for 1% of sales. It has threatened a large entity with a lawsuit, but no small ones.

U.S. Patent 4,555,775: The League describes AT&T's backup store patent as "Too Obvious to Publish." Yet, in a letter in this issue of *Communications,* Dennis Richie points out that this technology was published in the ACM [36] and was recently called "a seminal paper" whose ideas are seen in X Windows Macintosh and many other windows systems [14]. While AT&T has sent notification letters on this patent, it has put the patent into reexamination and has not threatened suit or sued anyone on this patent.

U.S. Patent 4,656,583: This is an IBM patent on compiler speedup.

U.S. Patent 4,742,450: This is an IBM-shared copy on write patent.

Why look for trouble you might never have to face? When and if a patent is brought to your attention you can decide what to do. If you do a search and find a patent, you might spend effort designing around a patent that its owner would never have asserted against you. If you do not design around it, you might be liable for treble damages because you were aware of the patent. Of course if you are competing against products protected by patents, you might want to check into their patents before you design your product, as you can expect your competitor to examine your product for infringement, thus you will probably have to face the problem one way or another. Here, it is good accounting practice to set aside a reserve for infringement.

2. Stay outside the claimed territory.

If the patent claims all the land west of the Mississippi and you stay on the east of the Mississippi, you will not infringe.

3. Go where people were.

If you know people were in Bismark before the patentholder landed in Left Fork, settling in Bismark will protect you. Distrust rumors about earlier settlers. Make sure the prior art is documented in a published paper or was obviously used in a product. If you ask around the industry you are likely to find pointers to prior art. You might want to send the prior art to the patent owner, or the PTO for insertion in the file wrapper. The file wrapper is a file containing all the correspondence on the patent with the PTO. Your patent lawyer might consult it to find prior art which might help you design around a patent or understand its scope.

4. Make a business deal with the patentholder.

Generally you can license, or cross-license a patent or find some other way to get rights.

5. Break the patent.

You can attempt to get the patent invalidated by proving it is invalid over the prior art, the disclosure was inadequate or it was otherwise invalid. This is risky and expensive where the patent is good and the patentholder determined.

A patent must claim something new, lest its owner usurp others' rights; it also must be on something nonobvious to prevent giving protection to insignificant improvements.

As technical people, we often look at a patent differently from the way entrepreneurs and judges do. We see Left Fork, North Dakota after it has become a thriving town, and are likely to say it is obvious—there are lots of places like Left Fork, and lots of them have similar buildings; thus constructing buildings in Left Fork seems obvious. It does not belong to the entrepreneur. Invite everyone!

The Left Fork patentholder, the entrepreneur, feels this is like arguing that it was obvious that land in Silicon Valley or Microsoft stock would appreciate in value. Given the advantage of hindsight, it is obvious, but the person who invested in the land, the stock, or the technology should benefit from its appreciation in value. The entrepreneur says, "I built the buildings based on my being granted rights to them and my having a vision of what I could make of it. And now you are looking for loopholes in my deed so others can move in! It may be a poor thing, but it is my own."

As technical people our immediate bias is to find inventions "obvious," because we focus on the technical sophistication, evaluated with the advantage of hindsight. The value of land, a patent or a copyright has to do with how the market evaluates it. If the land is in the Mohave desert; the copyrighted work banal, or the patent on technology people don't want, then it may be worthless. If the land is in downtown Manhattan, the copyright on Donald Duck, or the patent on technology others want, it can be very valuable. The important thing about an invention is not so much that it be inventive, but that it be new if it is to be patented, and that it be useful if people are to buy it.

These two patents are what IBM calls Group 1 patents whose royalty is 1% of sales. They have been licensed by IBM as part of general licensing agreements but have not been licensed individually. (About 50 of IBM patents are Group 2 patents. Group 2 patents can be licensed for 2% each; the entire Group 1 portfolio, for 2%; and the entire IBM patent portfolio for 5%.)

These two patents have not been litigated and I do not believe IBM has aggressively asserted these patents against anyone. IBM, like most companies, normally files for patents only to protect what they expect to become commercial products. We treat these patents as protecting commercial products.

Most patents are never asserted. Much of the value of patents, like that of the Swiss Army, is that they act as a deterrent. The patents described here are typical of the small number where the patentholder forces a resolution: the infringer may take a license, design around the patent, or produce prior art showing there was no infringement.

Many letters asserting patents are no trespassing signs, putting potential infringers on notice should they infringe or telling them not to. They require no action. The notified companies might send prior art back to the patentholder, who might send it to the PTO for reexamination. The "infringer" may ignore the notice, waiting to see the reexamined patent or for the patentholder to become more assertive. The resolution may be hidden, in that an infringer may design around the patent. Rarely, a product is withdrawn from the market. The statistics on the cited patents are summarized in Table 1.

Whether any patent including those described here, is valid and infringed is a complex legal and technical question. An advantage of the patent system is that the question is an objective one based on the patent, prior art, and the "infringing" device. Such a dispute is less acrimonious than one in which the task is to evaluate testimony where one person yells "thief," and other "liar." Whether infringement actually occurred in any of the cases is irrelevant. The relevant question is did the original patentholders bring commercial products to market based on the patented technology and motivated by the rights a patent bestows?

Trademarks: Apple paid $30 million to use the name "Apple"

Trademark law is like patent law in that the first one who claims it gets to own it. The Beatles recorded on their own label, Apple Records. When Apple Computer was founded it agreed not to use the *Apple* name in the music business. Later, when the Macintosh played music, the Beatles sued. Apple Computer settled, paying about $30 million dollars to use the name "Apple."

I have been publicly accused of extorting Apple. Did I extort Apple? Did the Beatles extort Apple? Should the computer business have its own *sui generis* trademark law?

Table 1.

Company Size	Large	Small	Total
Patent Activity			
Patents granted	4 100%	5 100%	9 100%
Protected commercial products	4 100%	5 100%	9 100%
License appears to be available	4 100%	4 80%	8 89%
Firm founded to develop technology	0 0%	4 80%	4 44%
Sued Large Entities	0 0%	4 80%	4 44%
Sued Small Entities	0 0%	2 50%	2 22%
First Suit against Small Entity	0 0%	0 0%	0 0%
Suits threatened	1 25%	5 100%	5 67%
Patent Asserted (Notice sent)	2 50%	5 100%	7 78%
Resolution (Patent Asserted)			
Infringement Removed	0 0%	1 20%	1 11%
Product removed from market	0 0%	1 20%	1 11%
Licenses	1 25%	2 40%	3 33%
Unresolved	1 25%	1 20%	2 22%
Nothing to Resolve (No notice)	2 50%	0 0%	2 22%
Total	4 100%	5 100%	9 100%

Patents cited in Against Software Patents. We treat multiple patents covering the same technology as a single patent. With the exception of Quantel, the small entities were less than a dozen people at the time the patent was filed, and the large entities were Fortune 1000 companies. We assume that if a lawsuit was threatened, a notice was sent; and if a lawsuit was filed, a lawsuit was first threatened. We assume that if a patent has been asserted—people have been sent notices—there is a matter to be resolved. Even if one characterizes Cadtrak and Refac as being in the business of litigating patents as some do, the relevant fact is that the original patentholders were small entities introducing commercial products protected by patents.

Analysis Results

The nine patents cited by the League summarized in Table 1 lead us to these conclusions:

1. Software patents stimulate companies to bring commercial products to market.

All nine patents protected commercial products.

2. Software patents stimulate new business formation.

Four of the nine patents were from startups founded to exploit the patented technology. A fifth filed for

its patent in its seventh year. All five companies struggled for years.

3. Software patents stimulate the commercial introduction of fundamental advances by small entities. The technology pioneered by at least three of the small patentholders was significant in that it started new product categories or were widely adopted in the industry.

4. Licenses are usually available where companies enforce patents. Only Quantel seems to be unwilling to license its patent.

5. Where similar-size companies had a dispute, they settled differences quickly without litigation. The only patent dispute between similar-size companies (XyQuest) was settled readily. No small entities were faced with a lawsuit brought by a large entity without the advantage of the patentholder having settled earlier with a large infringer.

6. Small entities incurred little if any royalty and litigation costs for infringing patents. The only disputes in which a small entity paid patent royalties or was sued were those in which the patentholder had previously settled disputes with larger companies. No case was cited in which a big company aggressively went after a small one over patents, unless a large company had respected the patents first. The only such instance the author knows of is IBM (see sidebar).

7. Patent piracy by large entities appears to be common and small entities have a tough time getting their rights respected. Four of the five small entities had large entities use their technology without first licensing it. *All four were forced to sue.* This makes it unlikely that all the patent disputes were an honest difference of opinion, although some probably were. For this reason "piracy" seems a fair characterization. These same small entities have had their patents mischaracterized and their motives inpugned in the academic, trade and business press read by the software community.

It can cost over a million dollars to litigate a patent through to trial. The data shows that large entities are quick to use their power to try to intimidate small ones into abandoning their rights or accepting nuisance settlements rather than address infringement issues on their merits. It appears that this high rate of patent piracy is caused in part by the *Federal Rules of Civil Procedure* which tilt the scales of justice against the weak.

Our results confirm the League's suggestion that big companies will readily bully small ones, but refutes its suggestion that a patentholder who asserts a patent will get showered with a gold. The yellow matter is not gold.

8. U.S. companies are slow to accept software innovations from outside sources. The Japanese adapt innovations from sources outside the company twice as fast as U.S. companies [31]. The technology protected by at least two of the patents (CadTrak and HyperRacks) was exposed to companies that later became the first infringers.

Japan's ability to accommodate outside [the firm] innovation may be one of the reasons it has been so

Big Companies Do Sue Small Ones

While the League says that big companies will use patents against small ones, it cites no example. Two came to my attention. (I was contacted because I had arguably relevant prior art on the first patent.) In both, IBM sued former employees to get ownership of patents on technology developed on their own time, unrelated to their work and only *after* the technology proved to have value in the market.

IBM vs. Goldwasser **Civil 5:91 00021 D. Conn.: IBM encouraged employees to develop software products on their own time and seek patent protection for them so IBM could evaluate them for marketing. Goldwasser developed such a software product, but IBM rejected it; he left IBM stating he intended to pursue his technology, as he did. Six years later, another company introduced a product that seemed to be infringing his patents and he sued them. That company claimed it was covered under its cross-license with IBM; IBM sued Goldwasser to get ownership of the patent.**

IBM vs. Zachariades **C-91 20419: Zachariades before and while working for IBM developed on his own time a plastic valuable to the medical industry. He kept IBM informed about what he was doing, applied for patents, started his own company, and licensed the technology to a medical prosthesis company. When he was not paid, he sued and a jury awarded him $99 million. IBM "suddenly" found out what was happening and fired, and sued, him for the patents, telling him they did it in part to "terrorize" other IBM employees.**

Companies like to hire litigators who know what it is like from the other side. In both cases IBM is represented by the same firm that represented Edwin Armstrong, the great inventor of modern radio when David Sarnoff and RCA were refusing to respect Armstrong's rights. Ken Burns tells the story in his PBS documentary, *Empire of the Air:* On January 31, 1954 Edwin Amstrong under the strain of RCA's tactics—well dressed as always, in a suit, overcoat, scarf and gloves—jumped from his 13th floor apartment onto the third story roof of the River Club below [29]. His widow won all the patent suits.

Having hired a firm that experienced firsthand the tactics that caused a great inventor to kill himself, IBM should be able to, by suing Goldwasser and Zachariades, "terrorize" its employees.

Table 2.

Innovation Source	Commercial Products	Efficacy (Patents Asserted)	Cost (000,000/yr)	Cost-effectiveness
Large entity	4	2	0.03	67.
Small entity	5	5	0.01	500.
Commercial Sector	9	7	0.04	175.
Federally funded	1	0	$487.0	0.0021

Cost Effectiveness to Taxpayers of Innovation Sources. None of the nine patents appears to result from federal funding. However, we arbitrarily allocate one patent to this category so as to prevent zero results. While the PTO is virtually self-funded, $1.8 million of its $419 million budget comes from taxpayers. Since about 25% of the patents are from small entities and 75% from large entities we distribute the $1.8 million accordingly. We assume that 2% of the patents are software related; it is probably less. Government R & D in computer science was $487 million in fiscal 1989 out of total federally funded research of $61 billion.

successful in dominating markets. If the U.S. is to exploit its strengths in innovation, it must learn to adapt outside innovations without the inefficiency of legal confrontation. Fast and efficient patent enforcement should encourage U.S. companies to license outside technology early rather than wait until they have an infringing product in the market and face legal exposure.

If large companies are forced to deal with infringement issue early, they might see it is to their advantage to work with the inventors, using their knowledge. Now, the legal system keeps the patentholder and infringer at war until such time as the patentholder's knowledge is of little value to the "infringer," thus wasting one of our most valuable resources—the creativity and experience of innovators.

9. Developers do not seem to be infringing multiple patents on a single product.

The only example that was cited in which someone faced infringement issues from more than one patentholder seems to be XWindows facing the CadTrak and AT&T patents, but this has not been resolved and no lawsuits seem to have been filed or threatened.

10. The patent system seems to reject bad patents early in the patent assertion process.

We think the League is right in alleging that bad patents have been issued. The League however, fails to identify a patent that was rejected by the courts. We think this is because issues of prior art and patent invalidity are considered early in the patent assertion process. Patentholders rarely continue to assert patents in the face of solid evidence of invalidity or noninfringement.

11. If software patents were more widely respected we would probably have had fewer variations on a theme, and more themes to vary on.

Product development effort seems to have focused on creating many versions of an invention once its value was proven. Over 250 different spreadsheets and at least four products generally considered to be HyperCard clones were marketed.

12. Big companies' patents do not seem to inhibit small developers.

The innovations protected by small entity patents listed here seem to have been more widely adopted than those of big companies in their industries. Big companies are better at commercializing and protecting their minor innovations, than their major ones.

That small entities seem to introduce the more fundamental innovations to the market is telling. Big companies are often unsuccessful in transforming innovations into commercial success: Xerox PARC pioneered much of modern-day personal computer and its software. Although IBM invented a predecessor to the spreadsheet (expired U.S. Patent 3,610,902), it did not market a commercial product based on it; it also did not assert the patent even though its claims seem to read on (i.e., be infringed by) modern spreadsheets. These technologies became major product categories primarily through the efforts of small entities.

13. Small entities using patents are exceptionally cost-effective in encouraging innovation—especially compared to federal funding.

Table 2 shows a rough estimate of the efficacy of three major sources of innovation: federally funded, large entity, and small entity. Our results show that small entities are 7.5 times as cost-effective at stimulating innovation as large ones, and 200,000 times as cost-effective as federal funding. The U.S. grants patent rights to universities as part of its research contracts; thus patents are issued in all these areas and patents asserted is a reasonable measure of innovation. We believe a more scientific study would refine these results, but doubt it would change the basic conclusion.

As a software developer, you might review the patents discussed and put yourself in the place of each of the parties involved. If your product finds satisfied users, do you think better financed companies with stronger marketing organizations will market competitive products, using your innovations? If so, will patents be useful to you? If a patent is enforced against you, do you think you will be able to design around it? If you have to license it, do you think your competitors will also have to license it, thus passing the cost on to the end customers? Which problem would you rather have: a big company entering a market you developed, or finding out you were accidentally infringing a patent? Do you think the effect of software patents might be more innovation, higher software prices and an industry with more long-term profitability?

If you are protected by patents, your success depends in part on your patented inventions as others must deal with them. If you acci-

dentally infringe a patent, designing around it is within your expertise. If you do not have patents, success depends much more on the ability to finance and market products—capabilities outside of your expertise and control. If you are a software developer, don't patents benefit you by manifesting your contributions in rights you can bring to the bargaining table, while confining the problems largely to your area of expertise and control? *Issues in Science and Technology* (Winter 1992) contains a letter from Commissioner of Patents Harry F. Manbeck who said of another article by the same authors that they [15]:

demonstrate they do not understand the current law . . . Most of their statements . . . do not appear to be the result of a balanced and reasoned inquiry and do not appear to be supported by the facts. . . . they cavalierly dismiss the view of those who appear to have used the patent system successfully and impugn their motives . . .

The PTO issued about 89,000 patents in 1990 from which the League, with the advantage of hindsight, can pick and choose the ones to attack. Consider the information presented here on patents the League selected to demonstrate the PTO's mistakes. Whose standards are higher, the League's or the PTO's?

Recommendations

After reviewing our results we can make some general recommendations.

1. Policy should be made on the assumption that innovation occurs in software as in other technologies until compelling evidence to the contrary is found.

This is consistent with the results described here. The operational implications are to continue to let the system operate as it is accepting evolutionary changes based on experience rather than speculation.

2. The PTO should be viewed as a source of innovation that competes for funding with other federally funded sources of innovation.

PTO fees should be reduced, especially for small entities, and the PTO should receive a higher level of funding to improve its ability to examine patents so it can issue better quality, more timely, patents in software and other technologies. European patent offices are much better equipped and much better funded. It seems that the PTO should compete with the NSF and other organizations for federal funding on the basis of their cost effectiveness in encouraging innovation.

In 1990 only $2 million of the PTO budget of $419 million came from federal funding; the remainder came from user fees. Superficially, it might seem that investing in the patent system will have a multiplier effect of 80,000 in creating innovation as compared to federally funded science. We suggest no such thing. We do however ask the question: If taxpayers were to spend an additional $160 million per year to support innovation, we could either increase the $64 billion federal funding on science by one-fourth of one percent (0.25%) or increase PTO funding by 40%, enabling the PTO to issue better patents and restore reduced user fees for small entities. Which will likely produce more innovation? Which will achieve a greater multiplier effect by encouraging additional private investment?

An example of federally funded science is fusion power research, which has been going on for at least 25 years, has cost hundreds of millions of dollars and has produced little practical result. Pons and Fleishman developed (and filed for patents on) cold fusion without government funding yet they, having invested their own money and not being in the mainline of governmental funding, are heavily criticized. While it is not clear that Pons and Fleishman have produced cold fusion, respected people in the field believe that they have, even if no one yet understands what is happening. This is just an example of how the system is biased in favor of government funding of expensive conventional solutions, and against individuals and small companies who risk their own time and money to innovate.

Individuals taking a contrary view have been the major source of new ideas in both science [38] and engineering [24]. This is why small entities and the patent system are so important. Most will fail, but the successes more than make up for the failures.

It would be interesting to evaluate the results of federally funded science to see which projects are worth the cost. Some projects may have become like those welfare mothers who, generation after generation, are entrapped in a governmental support system.

3. The patent laws should be modified to make it possible for small entities to assert their patent rights more effectively.

The data show it is commonplace for large companies to pirate the technology of small entities. No case was cited where a large company licensed a small entities' technology without first being sued suggests the existing laws do not motivate large companies to resolve patent disputes with small companies quickly. The issue here is not just fairness to inventors and improved efficiency in settling disputes. Rather, it is concerned with avoiding the waste that occurs because U.S. companies are so much slower at adopting new innovations than Japanese companies.

Congress responded with antipiracy legislation where software copyrights were concerned; we would hope it would similarly pass legislation to prevent patent piracy. Remedies similar to the criminal penalties for copyright infringement and Rule 11 sanctions for attorneys who file frivolous suits are worth considering. We suggest the following as possible remedies for patent disputes to stimulate discussion:

- After being put on notice, an "infringer" would have six months to

file any prior art to be used to defend the infringement suit with the PTO. (I find it difficult to believe it is well-known art if it can not be found in six months.)

- If a patentholder prevails in a lawsuit, the remedies should include an extension of the period of exclusivity against that infringer equal to the length of time the suit was in progress.
- Discovery should be limited.

These suggestions, which should induce speedier resolution of patents disputes, are suggested for all patents disputes, not just software.

The patent system, an enormously productive system for inducing innovation, is being stymied by a cumbersome dispute resolution process. Is it in the public good to have a system of conflict resolution that discourages conflict resolution? Should innovators spend their time innovating or litigating? If the courts could resolve software patent and copyrights issues more quickly, it would clarify the law so everyone can make decisions with some predictability. The problems are not unique to patents but occur in all litigation. That the judicial and even the legal, community are beginning to address the inefficiency of dispute resolution and litigation is grounds for cautious optimism.

4. Further study of the role of patents and federal funding in software innovation is useful.

We are keenly aware that the sample is small and unscientific, and thus our results should be considered suggestive rather than definitive. A more definitive study should be useful in bringing out facts that would be useful in evaluating future changes to the patent law.

These recommendations can be summarized thus: Redress the balance of incentives so innovators will prefer to develop their ideas commercially, using patent protection rather than search for federal funding.

The Software Patent Confrontation

The software industry is getting more competitive. Almost every company that has hit products uses its cash flow to develop entries in other product categories. As a result, product categories are getting very competitive. Since most software companies have confined their intellectual property to source code copyrights, user interface copyright and trademarks, whenever they come up with a successful innovation, their competitors will often quickly replicate it. As a result, the impetus is toward similarly featured products competing on price, differing only in the mistakes which the originators must maintain to support their existing customers.

Now companies are recognizing that by using patents they can compete on features and function—not just tactically, but strategically. Even if competitors do replicate the features, they will likely make them different enough to avoid infringement. Companies following this approach will support standards, but their products will have a substantial proprietary component engendering products with more diverse feature sets. This will enable the industry to compete more on the profitable playing field of unique capabilities and market position and less on price. This is consistent with standard business school product marketing, where product differentiation and market segmentation are basic.

Intellectual property has already driven the market for those who got in early and established standards. Lotus owns the 1-2-3 standard. Novell owns a major network standard and WordPerfect, a major word processor standard. Apple owns the Macintosh User interface standard and Intel and Microsoft own the IBM compatibility standard. Patents give new companies the opportunity to establish and own something of value in the market based on their innovativeness rather than their marketing and financial capabilities.

While the problem of people accidentally infringing software patents has been greatly exaggerated, several patents will be successfully asserted against existing products. This will be primarily between those companies that focused on innovation and have patents, and those that focused on exploiting recognized business opportunities. This kind of confrontation occurred earlier in the aircraft and other industries [21].

During these confrontations, the businesses with a large volume of infringing products will understandably, feel "extorted" since they did not anticipate patent infringement. Such businesspeople will take support for their position from those who argue against software patents and advocate or suggest invalidating existing software patents ([15, 39]).

The software innovators who advanced the technology and made business decisions based on their patent rights will similarly feel cheated especially where they pioneered commercial products based on their inventions. When depositors made decisions based on gov-

ALPHA: Abraham Lincoln Patent Holders Association

This organization, founded in January 1992, supports the use of, and educates people about, software patents. Already, ALPHA's members include two software patentholders whose patents have been litigated, four patentholders whose patents are mentioned here, two former board members of the Software Publishers Association and lawyers from Merchant and Gould, Baker and McKenzie, Welch and Katz and the Franklin Pierce Law Center, and a former commissioner of patents and trademarks.

ernment guarantees of S&L deposits, no one suggested that the government default on its obligations to insured depositors, as people suggest the government invalidate existing software patents. No one vilifies the S&L depositors because the government has to pay them money; yet software innovators find themselves vilified with lies and half truths.

In this confrontation, both sides start out feeling cheated.

Many "infringers" will react emotionally and view it as a problem to be gotten rid of and many will fight to the bitter end. This raises the stakes, since a company having been put on notice may be liable for treble damages and attorneys' fees. Patentholders will not likely pursue these cases for four or five years to let the infringers' liability build up. After a suit is filed these companies will be getting much of their advice from those who have most to profit from the litigation: their litigators. This seems to be what is happening to Lotus.

Some software developers on finding out that the rules were not what they thought, face the problems of infringing others' patents while not having patented their own successful innovations. Some will chalk it up as one of many risks and uncertainties of business. Those who react emotionally might find it useful to first ask: Which of the players have acted in good faith? Which have not? Which have been responsible for the patent mess? Which have been innocent victims? Having answered these questions, such developers can more effectively target their wrath.

Companies that act rationally will analyze the patents to ascertain their scope and validity, whether infringement is occurring, and how easy it is to remove the "infringing" capability. They will check with other licensees. They will consider the obvious options, such as taking out a license, removing the infringing capability, finding prior art and showing it to the patentholder, or fighting in court if that is the only possibility. They will probably try to address the problem early, before the liability builds and consider negotiating a license or using some form of alternative dispute resolution to resolve infringement and validity issues. If the problem is associated with purchased product, most companies will stand in back of their products and provide a license, warrant it against infringement, or provide guidelines on how to avoid infringement. It will put its "no problem" in writing.

Astute companies will view infringement as an opportunity in disguise. If the patent is good and competitors are, or soon will be, infringing it, the first licenses can generally get an inexpensive license forcing competitors to pay more if they want to use the technology. It may be possible to get an exclusive license on some feature which differentiates a product from the competitor's. It can be worthwhile to see if the patent covers useful capability which could be added to the product. The best time to negotiate for the license is when you do not have the liability of infringement but can offer to create a demand for that capability by incorporating it into a product. Invention being the mother of necessity, your competitors will be faced with the choice of paying a higher price to license the technology or leave it out, thus differentiating your product from theirs.

In brief, what superficially looks like another problem to be dealt with in the increasingly competitive, commodities-oriented software business, might prove to be what makes products less *price* competitive. Many industries have worked on this basis all along: patents make industries more diverse in their offerings, more profitable, more innovative, and ultimately will

What You Can Do

Insist that the issues be debated. Don't let one side present its case unchallenged. If you need literature to distribute or a developer to join a debate on software, contact ALPHA.

One way to help correct the misinformation about software patents is to join ALPHA, The Abraham Lincoln Patent Holders Association. ALPHA is trying to correct the misinformation on software patents and to provide a forum for people to deal with issues of software patents, such as how to avoid infringement, setting licensing fees, finding prior art. Contact ALPHA at:

ALPHA
146 Main St., Suite 404
Los Altos, CA 94022

You can help get the issue discussed intelligently in Congress. Tell Congress it is important that the issues be discussed in open hearings where everybody gets to present his or her side, hear the issues debated and make up their own mind. Write:

House Subcommittee on Intellectual Property
2137 Rayburn Bldg.
Washington, D.C. 20515

Senate Subcommittee on Patents, Trademarks and Copyrights
U.S. Senate
Washington, D.C. 20510

E.R. Kazenske
Executive Assistant to the Commissioner
Commission on Patent Reform
U.S. Patent and Trademark Office
Box 25
Washington, D.C. 20231

make the U.S. more competitive.

The essence of this article is simple: Software intellectual property issues are not inherently different in substance from other technologies; what motivates people is not inherently different; industry life cycle is not inherently different; marketing and business strategies and tactics are not inherently different; the law and policy issues are not inherently different; the technology is not inherently new. Software has been around for 40 years. The issues may be new to those who had no experience with them. But the only difference is that software is a mass market industry for the first time and real money is at stake.

Acknowledgments

I would like to thank Steve Lundberg, John P. Sumner, Susan Nycum, Lewis Gable, George Gates, David Pressman and Tom Hassing for their many useful comments.

References

1. American Bar Association, *Two Hundred Years of English and American Patent, Trademark and Copyright Law.* 1977.
2. Axelrod, R. *The Evolution of Cooperation,* Basic Books. 1985.
3. Brooks, F. No silver bullet: Essence and accidents of software engineering. *Computer* (Apr. 1987).
4. Bruce, R. *Lincoln and the Tools of War.* University of Illinois Press, 1989.
5. Bulkeley, W. Will software patents cramp creativity? *Wall Street J.* (Mar. 14, 1989).
6. Bugbee, B.W. *The Genesis of American Copyright Law.* Public Affairs Press, Wash., D.C., 1967.
7. CBEMA comments on computer-related invention patents. *Comput. Lawyer* (Oct. 1991).
8. Chisum, D. The patentability of algorithms. *U Pitt. L Review 47,* 959,971, (1986).
9. Chisum, D. *A Treatise on the Law of Patentability, Validity and Infringement,* (7 volumes) M. Bender, 1978.
10. Choate, P. *Agents of Influence.* Touchstone, 1990.
11. Clapes, A. Software copyright, and competition. *Quorem* (1989).
12. Cringely, R.X. *Accidental Empires.* Addison Wesley, 1991.
13. Fisher, L.M. Software industry in uproar over recent rush of patents. *New York Times,* May 12, 1989.
14. Foley, et al. *Computer Graphics: Principles and Practice,* Second ed. Addison Wesley, 1990.
15. Garfinkel, S., Stallman, R. and Kapor, M. Why patents are bad for software. *Issues in Science and Tech.* (Fall 1991).
16. Heckel, P. The Wright Brothers and Software Innovation, in *The Elements of Friendly Software Design,* Second ed., Sybex, 1991 (First ed. Warner Books, 1984).
17. Heckel, P. Isolating differences between files. *Commun. ACM* (Apr. 1978).
18. Heckel, P. Software patents and competitiveness. Op Ed, *San Francisco Examiner,* July 8, 1991, p. A-13.
19. Heckel, P. Zoomracks, designing a new software metaphor. *Dr. Dobbs J.* (Nov. 1985).
20. Heckel, P. and Lampson, B. A terminal oriented communications system. *Commun. ACM* (July 1977).
21. Heckel, P. and Schroeppel, R. Software techniques cram functions and data into pocket sized microprocessor applications, *Elect. Des.* (Apr. 12, 1980).
22. Howard, F. *Wilbur and Orville.* Knopf, 1988.
23. *In re Pardo,* 684 F.2d 912 (C.C.P.A. 1982).
24. Jewkes, J., Sawers, D. and Stillerman, R. *The Sources of Invention,* Second ed., Norton, 1969.
25. Judas, J.B. Innovation, A casualty at IBM, *Wall Street Jo.* (Oct. 17, 1991), A23.
26. Kahin, B. The software patent crisis. *Tech. Rev.* (Apr. 1960), 543–58.
27. League for Programming Freedom. Against software patents. *Commun. ACM* (Jan. 1992).
28. League for Programming Freedom. Software patents. *Dr. Dobbs J.* (Nov. 1990).
29. Lewis, T. Empire of the air. *HarperCollins* (1991).
30. Lincoln, A. *Selected Speeches and Writings,* Vintage, 1992.
31. Mansfield, E. Industrial innovation in Japan and the United States. *Science* (Sept. 30, 1988).
32. McCloskey, M. Intuitive physics. *Sci. Am.* (Apr. 1983), 123.
33. Neustadt, R.E. and May, E.R. Thinking in time: The uses of history for decisionmakers. *The Free Press,* 1986.
34. Nycum, S. Legal protection for computer programs. *Comput. Law J. 1,* 1 (1978).
35. Orwell, G. *Politics and the English Language.*
36. Pike, R. Graphics in overlapped bitmap layers. *ACM Trans. Graph. 17,* 3 (July 1983), 331.
37. Ritter, T. The politics of software patents. *Midnight Eng.* (May–June 1991).
38. Root-Bernstein, R. *Discovering.* Harvard University Press, 1989.
39. Samuelson, P. Benson revisited: The case against patent protection for algorithms and other computer program-related inventions, *Emory Law J. 39,* 1025, (1990).
40. Samuelson, P. CONTU revisited: The case against copyright program for computer programs in machine readable form. *Duke Law J. 663* (1984), 705–53.
41. Samuelson, P. Should program algorithms be patented? *Commun. ACM,* (Aug. 1990).
42. Samuelson, P. and Glushko, R. Survey on the look and feel lawsuit, *Commun. ACM* (May 1990).
43. Schon, D. Technology and Change, Delacorte, 1967.
44. Schwarzer, W. Science in the Courtroom. 15th Annual Intellectual Property Law Institute, Intellectual Property Section of the California State Bar, Nov. 1990.
45. Schwartz, E. The coming showdown over software patents. *Bus. Week* (May 13, 1991).
46. Sumner, J. and Lundberg, S. The versatility of software patent protection: From subroutines to look and feel. *Comput. Lawyer* (June 1986).
47. Sumner, J. and Lundberg, S. Software patents: Are they here to stay? *Comput. Lawyer* (Oct. 1991).
48. Slutsker, G. and Churbuck, D. Whose invention is it anyway? *Forbes* magazine (Aug. 19, 1991).

Unix is a registered trademark of Unix System Laboratories, Inc.

About the Author:

PAUL HECKEL has a small company that develops software. He also consults in the area of user interface design and has been used as an expert on software user interface intellectual property issues by the legal community. **Author's present address:** HyperRacks, Inc., 146 Main St., Suite 404, Los Altos, CA 94022.

How to monitor the new books, articles, and conference papers in computer science —

Subscribe to *Computing Reviews*

Each month, *Computing Reviews* brings you the experts' selection of the most significant new publications in the field of computing. To do so, they scan 450 journals, of which 300 are devoted to computer science, the proceedings of all the important computer science conferences held, and thousands of new books put out by academic, commercial, and society publishers around the world.

You can keep up with everything that's important for you to know about current computer literature by merely browsing through about 60 pages of carefully researched and tightly written reviews each month. For computer professionals, scientists, mathematicians, engineers, social scientists, educators, and librarians, ACM's *Computing Reviews* reduces information overload and cuts through the mass of new material that is published daily.

Subject areas covered include: general literature; hardware; computer systems organization; software; data; theory of computation; mathematics of computing; information systems; computing methodologies; algebraic manipulation; artificial intelligence; computer graphics; image processing; pattern recognition; simulation and modeling; text processing; computer applications; computing milieux.

Subscribe today – for the best way to monitor the literature on computing and information technology.

ISSN 0010-4884, ACM Order No. 10300
Monthly, $125.00; ACM members $34.00, Student members $29.00.

ACM Press, Publications Marketing, 1515 Broadway, New York, NY 10036-9998
To charge your order, call 1-800-342-6626

CALENDAR OF EVENTS

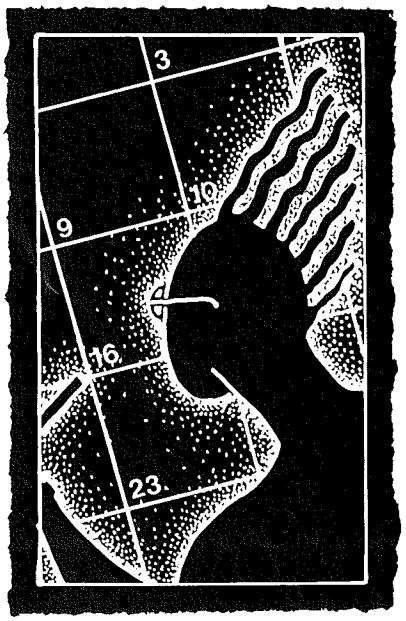
ACM's calendar policy is to list open computer science meetings that are sponsored by ACM, sister societies, or other scientific, technical or educational tax-exempt organizations. Educational seminars, institutes, and courses are not included due to space limitations. Listings for conferences NOT sponsored by ACM should include the title of the conference, sponsoring organization, a contact name and full address. Phone number, email address and/or fax number are optional. Please address to: Calendar Items, CACM, 1515 Broadway, New York, NY 10036; fax number: (212) 869-0481. For Conferences and Workshops sponsored or co-sponsored by ACM, the Calendar Listing should be included with the initial documents submitted for approval to ACM. All requests for ACM sponsorship or cooperation should be addressed to: CONFERENCE COORDINATOR, ACM Headquarters, 1515 Broadway, New York, NY 10036; (212) 626-0602; email: meetings at ACMVM (bitnet). The Technical Meeting Request Form (TMRF) for this purpose can be obtained from the Conference Coordinator. The TMRF should be submitted at least nine months in advance of the event to ensure time for processing the approval and to accommodate lead time for CACM listings.

The box in the upper left hand corner signifies the Conferences Board has given its approval for ACM sponsorship or cooperation. Conference Board Chair is Frank Friedman, Dept. of Computer and Information Sciences, Computer Center Building, Room 303, Temple University, Philadelphia, PA 19122; (215) 379-3945 (friedman;catemple.csnet).

June 17

ISSA 92: 1992 Conference and Expo on Information Systems Security

New York, NY. Sponsor: NY Metropolitan Chapter of the Information Systems Security Assoc. Contact Dione Levine, ISSA Conference, P.O. Box 1063, Grand Central Sta., New York, NY 10163-1063; (212) 242-3900.



June 17-19

SEKE '92: 4th International Conference on Software Engineering and Knowledge Engineering

Capri, Italy. Sponsor: Univ. of Salerno, Univ. of Naples, CRIAI, Univ. of Pittsburgh, SWIFT. Contact S.K. Chang, Dept. of Comp. Sci, Univ. of Pittsburgh, Pittsburgh, PA 15260; (412) 624-8432; email: chang@cs.pitt.edu.

June 17-19*

2d Workshop "Explication" of the PRC-GDR-IA (CNRS)

Sophia-Antpolis, France. Contact Danny Sergeant, INRIA Relations Exterieuries, BP 109, 06561 Valbonne, Cedex, France; (16) 93 65 77 05; fax: (16) 93 65 77 65; email: beahour@venera.isi.edu.

June 18-20

18th International Workshop on Graph-Theoretic Concepts in Computer Science

Wiesbaden-Naurod, Germany. Sponsor: Johann Wolgang Goethe Univ. Contact Ernst W. Mayr, J.W. Goethe Univ., Theoretische Informatik, Robert-Mayer-Str. 11-15, W-6000 Frankfurt am Main, Germany. ++49 69 7988325; email: mayr@thi.informatik.uni-frankfurt.de.

June 18-20

■ *TWLA 92: ACM Tutorials for Professional Development*

Los Angeles, Calif. Sponsor: ACM Local Activities Board, Los Angeles ACM and Orange County ACM Chapters. Contact David Oppenheim, 3507 Barry Ave., Los Angeles, CA 90066; (310) 397-2546; email: dave oppenheim@mts.cc.wayne.edu.

June 21-24

15th International ACM/SIGIR Conference on Research & Development in Information Retrieval

Copenhagen, Denmark. Sponsor: Royal School of Librarianship in coop. with SIGIR. Contact Peter Ingwersen, Danmarks Biblioteksskole, The Royal School of Librarian, 6 Birketinget, DK-2300 Copenhagen S; 45 31 58 60 66

June 22

■ *ACM/BPI Tech JOB Fair*

Columbia, Md. Sponsor: ACM and BPI. Contact Nick Harding, BPI, 187 Ballardvale St., Ste. 260, Wilmington, MA 01887; (508) 988-0881.

June 22-24

■ *LFP '92: ACM Conference on Lisp and Functional Programming*

San Francisco, Calif. Sponsor: SIGPLAN, SIGACT and SIGART. Contact Jon L. White, Lucid, Inc., 707 Laurel St., Menlo Park, CA 94025; (415) 329-8400; email: jonl@lucid.com.

June 22-24*

International Conference on Theorem Provers in Circuit Design: Theory, Practice and Experience

The Netherlands. Sponsor: IFIP TC10/WG 10.2, The Dutch National Facility for Informatics. Contact Huub van Thienen or Harriet Reker, Univ. of Nijmegen, Dept. of Comp. Sci., Computer and Communications Systems Group, P.O. Box 9010, NL 6500 GL Nijmegen, the Netherlands; email: germaine@cs.kun.nl.

June 22-25
7th Annual IEEE Conference on Structure in Complexity Theory
Boston, Mass. Sponsor: IEEE. Contact James Royer, School of Comp. and Info. Sci., Syracuse Univ., Syracuse, NY 13244; email: royer @top.cis.syr.edu.

June 22-25
Logic in Computer Science
Santa Cruz, Calif. Sponsor: IEEE TC. Contact Phokian Kolaitis, Comp. and Info. Sci., Univ. of California @ Santa Cruz, Santa Cruz, CA 95064; email: kolaitis@saturn.ucsc.edu.

June 22-26
CGI '92: Computer Graphics International '92
Tokyo, Japan. Sponsor: Kogakuin Univ. Contact T. Takahashi, CGI92, Autonomous Robot Systems Lab, NTT Human Interface Labs, 3-9-11 Midori-cho, Musashino-shi, Tokyo 180, Japan; 81 422 59 2406; fax: 81 422 59 2245; email: cgi92@nttarm.ntt.jp.

June 29-July 1
4th Workshop on Computer-Aided Verification
Montreal, Quebec, Canada. Sponsors: Concordia Univ. et al. Contact Gregor Von Bochman, Univ. de Montreal, Dept. d'IRO, 2900, Edouard-Montpetit Blvd., V-240, Montreal, Quebec, H3C 3J7 Canada; (514) 343-7484; fax: (514) 343-5834; email: bochman@iro.montreal.ca.

June 29-July 1
ATABLE-92:2d International Workshop on Array Structures
Montreal, Quebec, Canada. For sponsorship and information contact Martine Gemme, ATABLE-92, DIRO Univ. de Montreal, P.O. Box 6128, Sta. A, Montreal, Quebec, Canada H3C 3J7; (514) 343-6602; fax: (514) 343-5834; email: gemme@irp.umontreal.ca.

June 29-July 3
ECOOP '92, 6th European Conference on Object-Oriented Programming
Utrecht, the Netherlands. Contact Gert Florjin, Software Engineering Research Centre, P.O. Box 424, 3500 AK Utrecht, the Netherlands; +31 (30) 322640; fax: +31 (30) 341249; email: ecoop92@serc.nl.

July 6-10
■ *APL '92: International Conference on APL*
St. Petersburg, Russia. Sponsor: SIGAPL, SovAPL, FinnAPL, USSR Academy of Science. Contact Erkki Juvonen, Finnish APL Association Simontie 6 D 126, 01350 Vantaa, Finland; or US Contact, Lynne Shaw, 417 Riverside Drive, #9B, New York, NY 10025; (212) 662-2400.

July 6-10
4th International Conference on Information Processing and Management Uncertainty in Knowledge-Based Systems
Palma de Malorca, Spain. Contact IPMU '92 Conference, c/o L. Valerde, Dept. of Math i Info. Univ. Illes Balears, Carretera de Valldemossa, Km 7,5.070701, Palma de Mallorca, Spain; fax: 34 71 758061 email: dmilvg0@ps.uib.es.

July 6-10
3d Forum on Informatics
Hanoi, Viet Nam. Sponsor: Viet Nam Assoc. for Information Processing. Contact the assoc. at 36C Ly Nam De, Ha-noi, Viet Nam; +84 4 254219; fax: +84 4 256849.

July 7-9
ICCHP '92: International Conference on Computers for Handicapped Persons
Vienna, Austria. OCG, Tech. Univ. of Vienna, GI, SI, TermNet. Contact Wolgang Zagler, Tech. Univ. of Vienna, Inst. for Electronics, Gusshausstreabe 27/359/1B, A-1040, Vienna, Austria; +43 1 58801-3887; fax: +43 1 505 26 66; email: zw@fortec.tuwien.ac.at.

July 8-10
FTCS-22: International Symposium on Fault-tolerant Computing
Boston, Mass. Sponsor: UMASS @ Amherst. Contact Dhiraj Pradham, Symposium Chair, Dept. of Electrical and Computer Eng., Univ. of Massachusetts, Amherst, MA 01003; (413) 545-0160; fax: (413) 545-4611; email: pradham@ecs.umass.edu.

July 12-17
AAAI-92, 10th National Conference on Artificial Intelligence and 4th Annual Conference on Innovative Applications of Artificial Intelligence
San Jose, Calif. Sponsor: American Association for Artificial Intelligence. Contact AAAI-92, 445 Burgess Drive, Menlo Park, CA 94025; (415) 328-3123; fax: (415) 321-4457; email: iaai@aaai.org or ncai@aaai.org.

July 13-17
■ *State of the Art in Computer Graphics*
Reading, UK. Sponsor: SIGGRAPH and British Computer Society. Contact Rae Earnshaw, +44 532 335414; email: eclrae@leeds.cmsl.

July 19-24
■ *ICS '92: ACM SIGARCH International Conference on Supercomputing*
Washington, DC. Sponsor: SIGARCH. Contact Ken Kennedy, CITI Rice Univ., P.O. Box 1892, Houston, TX 77251; (713) 527-6009; email: ken@rice.edu.

July 20-24
■ *Logic at Tver '92: Logical Foundation of Computer Science*
Tver, Russia. Sponsor: Math Science Inst., IEEE, EATCS, ASL, in coop. SIGACT. Contact M.A. Taitslyn, Univ. of Tver, 33 Zhelyahova St., Tver 170013 Russia.

July 23-25*
1st Annual Conference on Multimedia in Education & Industry—Training for the 21st Century
Charleston, S.C. For sponsorship and additional information, contact Ronald D. Plemmons, S.C. Board for Technical and Comprehensive Education, 111 Executive Dr., Columbia, SC 29210; (800) 553-7702.

July 27-29
■ *ISSAC: International Symposium on Symbolic Algebraic Computation*
Berkeley, Calif. Sponsor: SIGSAM. Contact Erich Kaltolfen, Rensselaer Polytech. Inst., Dept. of Comp. Sci., Troy, NY 12180; (518) 276-6907; email: kaltolfen@cs.rpi.edu.

July 27-29
■ *5th Annual Workshop on Computational Learning Theory*
Pittsburgh, Penn. Sponsor: SIGACT and SIGART. Contact Robert Daley, Univ. of Pittsburgh, Dept. of Comp. Sci., Pittsburgh, PA 15260; (412) 624-5930; email: daley@cs.pitt.edu.

July 27-31
■ *SIGGRAPH '92: 19th International Conference on Computer Graphics and Interactive Techniques*
Chicago, Ill. Sponsor: SIGGRAPH. Contact Smith, Bucklin, & Assoc., 401 N. Michigan Ave., Chicago, IL 60611; (312) 644-6610.

August 4-7
■ *The St. Petersburg International Workshop on Human-Computer Interaction*
St. Petersburg, USSR. Sponsor: ICTSI in coop.w/SIGCHI. Contact Larry Press, California State Univ., Comp. Sci. Dept., 10726 Esther Ave., Los Angeles, CA 90064; (310) 675-6515; email: lpress@nenera.isi.edu.

August 5-6
Fed Micro '92
Washington, DC. Sponsor: National Trade Productions. Contact Sylvia Griffiths, (800) 638-8510 or (703) 683-8500.

August 9-14
AAAI's Tenth Annual Conference on AI (AAAI-92)
San Diego, Calif. Sponsor: The American Association for Artificial Intelligence. Contact AAAI-92, Burgess Drive, Menlo Park, CA 94025-3496; (415) 328-3123.

August 10-12
■ *PODC '92: 11th Annual ACM Symposium in Principles of Distributed Computing*
Vancouver, BC, Canada. Sponsor: SIGOPS and SIGACT. Contact Norm Hutchinson, Univ. of British Columbia, Dept. of Comp. Sci., 6356 Agriculture Rd, Vancouver, BC, Canada V6T 1Z2; (604) 822-8188; email: hutchinson@cs.ubc.ca.

August 10-14*
LUV-92, Lisp Users Vendors Conference
San Diego, Calif. For sponsorship and information contact Laura Lotz, An Event to Remember, P.O. Box 294, Malvern, PA 19355-0294; (215) 651-2990; fax: (215) 651-0936 or Jim Argones GE-CRD K1-5c30, P.O. Box 8, Schenectady, NY 12301; (518) 387-6967; email: jka@vega.crd.ge.com.

August 14-16
7th International Conference on Computing and Philosophy
Orlando, Fla. Sponsor: The Committee on Computer Use in Philosophy of the American Philosophical Association, Dept. of Philosophy and Humanities, Dept. of Comp. Sci., and the Journal, Philosophy and Computing. Contact Don Jones, Philosophy, UCF, Orlando, FL 32816; email: asdihifaa@ucf1vm (internet: asdihifaa@ucf1vm.cc.ucf.edu.).

INDEX TO ADVERTISERS

JUNE 1992

For further information regarding advertising, call the advertising representative in your area :

Northeast/Midwest/South:
Cascio Communications
(312)755-0020

West:
Marshall Rubin & Associates
(818)782-1511

All foreign:
ACM Headquarters
(212)869-7440

New England/Mid-Atlantic/West:
Recruitment Display Advertising
Community Advertising Service
(201)487-2511

Advertising Associates
(201)461-5422

FREE PRODUCT INFORMATION

Choose from three simple ways to obtain FREE product information.

1. Circle the # on the Reader Service Card that corresponds to the "circle number" on the advertisement or the Advertiser's Index.

2. For *quick* response, write the Reader Service Card number in the space provided on the RSC card for a sales rep to contact you.

3. For *IMMEDIATE* response, use the telephone number listed for the advertiser in the Advertiser's Index on the opposite page and call the company directly.

GET THE COMPETITIVE EDGE THE EASY WAY-

use these cards to request information about the products and services advertised in this month's issue. No cost! No obligation!

Please print

Name_____

Title_____

Company_____

Address_____

City/State/Zip_____

Telephone_____ FAX_____

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

How do you receive your copy of *Communications?*

☐ Membership ☐ Newsstand

Primary activity of your organization:
☐ 1–Business/industry other than computing
☐ 2–Computing ☐ 3–Education
☐ 4–Government ☐ 5–Reseller/VAR
☐ 6–Other _____

Plan to buy:
☐ 7–Within 2 months ☐ 8–3-6 months
☐ 9–7-12 months ☐ 10–Beyond 12 months
☐ 11–Reference only

Need RUSH service? Use this "hot line" to have a sales rep contact you immediately (write in RSC numbers):
_____ # _____ # _____

Your comments on *Communications'* contents_____

MAIL TODAY! It's postage free. Act now to get the competitive edge.
Please note this card expires September 30, 1992.

Feedback on the magazine:
June 1992

Please rate this month's contents. The editors of *Communications* are interested to know whether the information in the articles or departments was useful to you. Please circle Y (yes) or N (no).

Y	N	Visualization in Networked Environments
Y	N	Metacomputing
Y	N	The Distributed Laboratory: An Interactive Visualization Environment for Electron Microscopy and 3D Imaging
Y	N	The CAVE – Audio Visual Experience Automatic Virtual Environment
Y	N	Modeling and Analysis of Empirical Data in Collaborative Environments
Y	N	An Object-Oriented Methodology for Knowledge/Base Database Coupling
Y	N	Parallel Database Systems: The Future of High Performance Database Systems
Y	N	Bit-Tree: A Data Structure for Fast File Processing

Please print

Name_____

Title_____

Company_____

Address_____

City/State/Zip_____

Telephone_____ FAX_____

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

How do you receive your copy of *Communications?*

☐ Membership ☐ Newsstand

Primary activity of your organization:
☐ 1–Business/industry other than computing
☐ 2–Computing ☐ 3–Education
☐ 4–Government ☐ 5–Reseller/VAR
☐ 6–Other _____

Plan to buy:
☐ 7–Within 2 months ☐ 8–3-6 months
☐ 9–7-12 months ☐ 10–Beyond 12 months
☐ 11–Reference only

Need RUSH service? Use this "hot line" to have a sales rep contact you immediately (write in RSC numbers):
_____ # _____ # _____

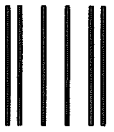
Your comments on *Communications'* contents_____

MAIL TODAY! It's postage free. Act now to get the competitive edge.
Please note this card expires September 30, 1992.

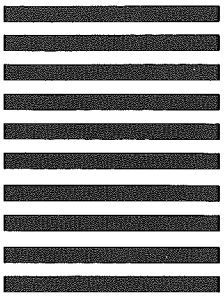
Feedback on the magazine:
June 1992

Please rate this month's contents. The editors of *Communications* are interested to know whether the information in the articles or departments was useful to you. Please circle Y (yes) or N (no).

Y	N	Visualization in Networked Environments
Y	N	Metacomputing
Y	N	The Distributed Laboratory: An Interactive Visualization Environment for Electron Microscopy and 3D Imaging
Y	N	The CAVE – Audio Visual Experience Automatic Virtual Environment
Y	N	Modeling and Analysis of Empirical Data in Collaborative Environments
Y	N	An Object-Oriented Methodology for Knowledge/Base Database Coupling
Y	N	Parallel Database Systems: The Future of High Performance Database Systems
Y	N	Bit-Tree: A Data Structure for Fast File Processing



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



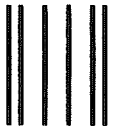
BUSINESS REPLY MAIL

First Class Permit No. 780 Pittsfield, MA

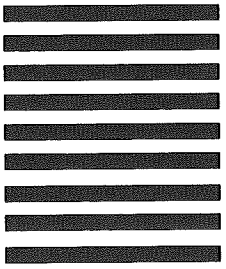
POSTAGE WILL BE PAID BY ADDRESSEE

COMMUNICATIONS OF THE ACM

Free Product Information
P.O. Box 5351
Pittsfield, MA 01203-9887



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL

First Class Permit No. 780 Pittsfield, MA

POSTAGE WILL BE PAID BY ADDRESSEE

COMMUNICATIONS OF THE ACM

Free Product Information
P.O. Box 5351
Pittsfield, MA 01203-9887

August 17-20
■ *SIGCOMM Symposium, Communication, Architectures, and Protocols*
Baltimore, Md. Sponsor: SIGCOMM. Deepinder Sidhu, Univ. of Maryland, Baltimore County Campus, Comp. Sci. Dept., 5401 Wilkins Ave., Baltimore, MD 21264; (301) 455-3028; email: sidhu@umbc3.umbc.edu.

August 17-21*
21st International Conference on Parallel Processing
Charles, Ill. For sponsorship and additional information contact T. Feng, E.E. East Bldg., The Pennsylvania State Univ., University Park, PA 16802; (814) 863-1469; fax: (814) 865-7065; email: tfeng@ecl.psu.edu.

August 19-21
■ *International Workshop on Distributed Object Management*
Edmonton, Alberta, Canada. Sponsor: Univ. of Alberta in coop. w/SIGMOD. Contact M. Tamer Ozsu, Univ. of Alberta, T6G 2H1 Canada; (403) 492-4589; email: ozsu@cs.alberta.ca.

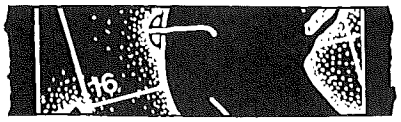
August 24-27
■ *18th International Conference on Very Large Data Bases*
Vancouver, BC, Canada. Sponsor: CIPS and VLDB Endowment in coop. with SIGMOD. Contact Paul Sorenson, Univ. at Alberta, 615 General Services Bldg., Edmonton, Alberta, T6G 2HI Canada; (403) 492-4589; email: sorenson@cs.ualberta.ca.

August 24-27
CONCUR '92
Stony Brook, N.Y. Contact Scott Smolka, Dept. of Comp. Sci., SUNY@Stony Brook, Stony Brook, NY 11794; (516) 632-8453; fax: (516) 632-8453; email: sas@cs.sunysb.edu.

August 26-28
4th International Symposium on Programming Language Implementation and Logic Programming (PLILP 92)
Leveun, Belgium. For sponsorship and information contact Maurice Bruynooghe, Dept. of Comp. Sci., Katholieke Univ. Leuven, Celestijnenlaan 200A, B-3001, Heverlee, Belgium; fax: ++32-(0) 16-20.53.08; email: maurice@cs.kuleuven.ac.be.

August 31-September 2
FPL 92: International Workshop on Field-Programmable Logic and Applications
Vienna, Austria. Sponsor: Vienna Univ. of Technology and Univ. Kaiserslautern. Contact Herbert Grunbacher, FPL 92, Vienna Univ. of Technology, Treitlstrasse 3, A-1040 Vienna, Austria; +43 1 58801-8150; fax: +43 1 569697; email: herbert@vlsivie.tuwien.ac.at.

September 2-4
3d International Workshop on VLSI for Artificial Intelligence and Neural Networks
Oxford, England. Sponsor: Univ. of Oxford. Contact Jose G. Delgado-Frias, Dept. of Electrical Engineering, SUNY@Binghamton, NY 13902-6000; (607) 777-4806; fax: (607) 777-4822; email: delgado@bingvaxu.cc.binghamton.edu.



September 2-4
DEXA 92: 3d Conference on Database and Expert Systems and Applications
Valencia, Spain. Sponsor: Tech. Univ. of Valencia, Research Inst. for Applied Knowledge Engineering, Austrian Comp. Soc., and German Comp. Soc. Contact Roland Wagner, Inst. of Comp. Sci., Univ. of Linz, A-4040 Linz, Austria; +43 (732) 2468-791; fax: +43 (732) 243989; email: a4423dab@awiuni11.bitnet.

September 7-8
2d European Workshop on Software Process Technology
Trondheim, Norway. Sponsor: AFCET, AICA, Norwegian Computer Society, et al. Contact Jean Claude Derniame, Centre de Recherche en Informatique de Nancy, CRIN, Campus Scientifique, B.P.O. 239, F-54506 Vandoeuvre Les Nancy, France; +33 83.413052; fax: +33 83.413079; email: derniame@loria.crin.fr.

September 7-10
■ *European Design Automation Conference*
Hamburg, Germany. Sponsor: SIGDa, GI, ITG, GME, BCS, NGI, AFCET, IEEE CS, IFIP. Contact Egon Horbst, Siemens AG ZFE IS EA, Postfach 830953, D-8000 Munchen 83; 49 089 636 3354; email: mcvax!unido!ztivax!hoerbst.

September 7-11
IFIP Congress 1992: 12th World Computer Congress
Madrid, Spain. Sponsor: International Federation for Information Processing. Contact Grupo Geyseco, IFIP '92, Mauricio Legendre 4, BG, E-28046 Madrid, Spain; fax: (+34-1) 3234936; email: ifip@dit.upm.es.

September 8-9
2d European Modula-2 Conference
Leicester, England. Sponsor: Leicester Polytechnic. Contact Sue Brookes, Modula-2, Leicester Polytechnic, Marketing Centre, P.O. Box 143, Leicester LE1 9BH England; (0533) 577098; fax: (0533) 549972.

September 8-10
MDBD-92: Baltic Conference on the Methods of Database Design
Riga, Latvian Republic. Sponsor: FRAME, Ltd. in coop. Baltic Coop. Council, ICM Univ, of Stockholm, and Latvian Academy of Sciences. Contact Boris Cadish, MDBD-92, Perses St., 2, Riga, Latvia, 226400; 211510; fax: (0132) 282524.

September 14-16
DCCA-3, 3d IFIP Working Conference on Dependable Computing for Critical Applications
Mondello (Palermo), Sicily, Italy. Sponsor: IFIP Working Group 10.4. Contact Luca Simoncini, Dipartimento Di Ingegneria dell'Informazione, Univ. of Pisa, Via Diotisalvi 2, 56100 Pisa, Italy; +(39) 50 593443 or 550100; fax: +(39) 554342; email: simon@icnucevm.cnuce.cnr.it.

September 15-18*
■ *HCI '92 People and Computers*
York, United Kingdom. Sponsor: BCS, HCI Group, in coop. w/SIGCHI. Contact Francoise Vassie, Center for Continuing Education, Univ. of York, York, Y01 ZEP, United Kingdom; 011 44 904 433949; email: fvi@uk.ac.york.

September 20-23*
■ *7th Annual Knowledge Based Software*
Tysons Corner, Va Sponsor: Rome Lab. in coop.w/SIGART and SIGSOFT. Contact W. Lewis Johnson, USC/Information Sciences Institute, 4676 Admiralty Way, Marina del Ray, CA 90292-6695; (310) 822-1511; email: johnson@isi.edu.

September 23-25*
■ *International Conference on Data Transmission*
London, United Kingdom. Sponsor: SIGCOMM, SIGOIS, and IEE. Contact Jane Chopping, Inst. of Electrical Engineers, Savoy Place, London WC2R 0BL United Kingdom; 071 240 1871.

September 23-25
■ *PRICAI 92: Pacific Rim International Conference on Artificial Intelligence*
Seoul, Korea. Sponsor: KISS, CAIR in coop. w/SIGART. Contact Jin Hyung Kim, Korea Advanced Inst. of Sci. & Tech., 373-1 Kusong-dong Yusong-ku; 82 42 829 3517; email: jkim@cair.kaist.ac.kr.

September 24-25
International Workshop on Object Orientation in Operating Systems IWOOOs '92
Paris, France. Sponsor: Inst. National Recherche en Informatique et Automatique, INRIA, IEEE Technical Workshop on Operating Systems and Application Environments. Contact Roy Campbell, Univ. of Illinois, Dept. of Comp. Sci., 2413 Digital Lab, 1304 W. Springfield Ave., Urbana, IL 61801; (217) 333-3328; email: roy@uiuc.edu.

September 29-October 1
EUROSIM '92: Eurosim Simulation Congress
Capri, Italy. Sponsor: SCSI, CASS, CSSC, CNR Italy. Contact A. DiChiara, Dept. of Civil Engineering, Univ. of Rome "Tor Vergata", via della Ricerca Scientifica, I-00173 Roma, Italy; +39 6 72594575; fax: +39 6 72594586.

September 30-October 2
■ *International Workshop on Hardware-Software Codesign*
Estes Park, Co. Sponsor: SIGDA, SIGSOFT, IEEE-CS, and IEEE-C&CS. Contact Joanne Degroat, Ohio State Univ., 205 Neil Ave., Columbus, OH 43210; email: degroat@ee.eng.ohio-state.edu.

September 30-October 2
13th Annual Allerton Conference on Communication, Control and Computing
Monticello, Ill. Sponsor: Univ. of Illinois at Urbana-Champaign. Contact P. Van Dooren, (217) 333-0656; email: vdooren@uics1.uiuc.edu or M. Spong (217) 333-4281; email: spong@lagrange.cs1.uiuc.edu.

October 5-7
■ *CSEE 92: 6th SEI Conference on Software Engineering Education*
San Diego, Calif. Sponsor: SEI in coop w/IEEE-CS, SIGCSE, SIGSOFT. Contact Carol Sledge, Software Engineering Institute, Rm 4206, 4500 5th Ave., Pittsburgh, PA 15213; (412) 268-7708; email: cas@sei.cmu.edu.

October 5-7
11th Symposium on Reliable Distributed Systems
Houston, Tex. Sponsor: IEEE Computer Society TC on Distributed Processing and TC on Fault-Tolerant Computing, IFIP WG 10.4 on Dependable Computing. Contact Kishor S. Trivedi, Dept. of Electrical Engineering, Duke Univ., Durham, NC 27706; (919) 660-5269; email: kst@egr.duke.edu.

October 5-7
2d International Conference On Software Quality Control
Research Triangle Park, N.C. Sponsor: American Society for Quality Control Software Division. Contact Sue McGrath, SAS Inst., SAS Campus Dr., Cary, NC 27513; email: sassam@dev.sas.com or John E. Lowe, Litton Computer Services, 4020 Executive Dr., Dayton, OH 45430; (513) 429-6458.

October 5-7
International Workshop Compiler Construction
Paderborn, Germany. Sponsor: German Gesellschaft fur Informatik, IFIP WG 2.4, GI Fachgruppe 2.1.3. Contact P. Pfahler, Univ.-GH Paderborn, Fachberiech Mathematick/Informatik, Warburger Str. 100, 4790 Paderborn, Germany; 05251 603069; fax: 05251 603836; email: peter@uni-paderborn.de.

October 7-9
3d International Symposium on Software Reliability Engineering
Research Triangle Park, N.C. Sponsor: IEEE CS, Alcatel Network Systems, BNR, Fujitsu, Northern Telecom, Storage Technology Corp. Contact M.A. Vouk, North Carolina State Univ., Dept. of Comp. Sci., Box 8206, Raleigh, NC 27695; (919) 515-7886; email: vouk@adm.csc.ncsu.edu.

October 12-14
3d International BANKAI Workshop: 1992 Adaptive Intelligent Systems
Brussels, Belgium. Sponsor: SWIFT. Contact Bob Phelps, A.I. Labs, 1 Avenue Adele, 1310 La Hulpe, Belgium; +32 2 655 31 53; fax: +32 2 655 3785.

October 12-15
■ *ASPLOS V: 5th Architectural Support for Programming Languages and Systems*
Boston, Mass. Sponsor: SIGARCH, SIGOPS and SIGPLAN. Contact Barry Flahive, Hewlett-Packard Apollo Systems, 300 Apollo Drive, MS CHR 02 DE, Chelmsford, MA 01824; (508) 256-6600; email: flahive@apollo.hp.com.

October 13-16
■ *SIGDOC '92: 10th Annual ACM Conference on Systems Documentation*
Ottawa, Canada. Sponsor: SIGDOC. Contact Roy MacLean, Bell Northern Research, P.O. Box 3511, Sta. C, Ottawa, Ontario, Canada. K1Y 4H7; (613) 763-4896; email: maclean@bnr.ca.

October 18-22
■ *OOPSLA '92: Object Oriented Programming Systems, Languages, Architectures*
Vancouver, Canada. Sponsor: SIGPLAN. Contact John Pugh, School of Comp. Sci., Carleton Univ., Colonel By Drive, Ottawa, Ont., Canada K1S 5B6; (613) 788-4330; email: john pugh@Carleton.ca.

October 19-20
■ *1992 Volume Visualization Workshop*
Boston, Mass. Sponsor: SIGGRAPH. Contact Larry Gelberg, Stardent 6, New England Technical Center, 521 Virginia Rd., Concord, MA 01742; (508) 287-0100; email: lar ryg@stardent.com.

October 19-21
Frontiers '92: 4th Symposium on the Frontiers of Massively Parallel Computation
McLean, Va. For sponsorship and more information contact Pearl Wang, Comp. Sci. Dept., George Mason Univ. Fairfax, VA 22030-4444; (703) 993-1530; email: f92info@gmuvax2.gmu.edu.

October 19-23
4th International Workshop on Foundations of Models and Languages for Data and Objects: MODELLING DATABASE DYNAMICS
Volkse, Germany. Sponsor: GI Working Group Foundation of Information Systems, in coop. with EATCS. Contact Udo Lipeck, Inst. fur Informatik, Univ. Hannover, Lange Laube 22, D-W 3000 Hanover 1, Germany; 49 (511) 762-4950; email: ul@informatik.uni-hannover.de.

October 19-23
Visualization '92
Boston, Mass. Sponsor: IEEE CS. Contact IEEE CS, Conference Dept., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903; (202) 371-1013.

October 25
3d ASIS SIG/CR Classification Research Workshop
Pittsburgh, Penn. Sponsor: Special Interest Group on Classification Research of the American Society for Information Science. Contact Raya Fidel, Graduate School Library and Information Science, Univ. of Washington, FM-30, Seattle, WA 98195; (206) 543-1888; fax: (206) 685-8049; email: fidel@u.washington.edu.

October 26-29
3d International Conference on Principle of Knowledge Representation and Reasoning (KR '92)
Cambridge, Mass. Sponsor: AAAI and CSCSI in coop. w/ECCAI and IJCAII. Contact William Swartout, USC/Information Sciences Inst., 4676 Admiralty Way, Marina del Rey, CA 90292-6695; (213) 822-1511; fax: (213) 823-6714; email: swartout@isi.edu.

October 27-30
3d Eurographics Workshop on Object-Oriented Graphics
Switzerland. Sponsors: Eurographics, C.U.I. Contact X. Pintado, C.U.I., Univ. of Geneva, 12 rue du Lac, 1207 Geneva, Switzerland; +41 22 787 6586; fax: +41 22 735 3905; email: eoog@cui.unige.ch.

November 1-4
■ *CSCW '92: Computer Supported Cooperative Work '92*
Toronto, Ontario, Canada. Sponsor: SIGCHI and SIGOIS. Contact Marilyn Mantei, CSRI, Rm. 398, Pratt Building, 6 King's College Rd., Univ. of Toronto, Toronto, Ontario, Canada M5S 1A4; (416) 978-5512; email: mantei@dgp.toronto.edu.

November 1-4
ORSA/TIMS National Joint Meeting
San Francisco, Calif. Sponsor: ORSA and TIMS. Contact Chaiho Kim, The Leavey School of Business, Univ. of Santa Clara, CA 94035.

November 3-6
3d European Conference on Software Quality
Madrid, Spain. Sponsor: EOQ Software Committee. Contact Julio Gonzalez Sanz, AECC-CONGHRISA, c/o Velazquez 90 5, 28006 Madrid (SPAIN); +34 1 575 25 80; fax +34 1 577 38 74.

November 4-6
1992 IEEE International Workshop on Defect Tolerance VLSI Systems
Dallas, Tex. Sponsor: IEEE. Contact F. Lombardi, Dept. of Comp. Sci., Texas A&M Univ., College Station, TX 77843-3112; (409) 845-8641; fax: (409) 847-8578; email: lombardi@cs.tamu.edu.

November 6-7
PDC '92—Participatory Design Conference
Cambridge, Mass. Sponsor: Computer Professionals for Social Responsibility. Contact Michael J. Muller/PDC'92, Bellcore RRC-1H229, 444 Hoes La., Piscataway, NJ 08854; (908) 699-4892; fax: (908) 336-2932; email: michael@bellcore.com.

November 8-12
■ *IEEE International Conference on Computer Aided Design*
Santa Clara, Calif. Sponsor: SIGDA, IEEE CAS, IEEE CS. Contact Louise Trevillyan, IBM Corp., P.O. Box 218, MS 33-145, Yorktown Heights, NY 10598; (914) 945-1507; email: louise@watson.ibm.com.

November 9-12*
■ *Conference on Software Maintenance '92*
Orlando, Fla. Sponsor: SIGSOFT and IEEE CS. Contact Vaclav Rajlich, Wayne State Univ., Dept. of Comp. Sci., Detroit, MI 48202; (313) 577-5423; email: vtr@cs.wayne.edu.

November 12-13
2d Workshop on the Management of Replicated Data
Monterey, Calif. Sponsor: IEEE TCOS. Contact Jehan-Francois Paris, Dept. of Computer Science, University of Houston, Houston, TX 77204-3475; (713) 749-3943; fax: (713) 749-2378; email: paris@cs.uh.edu.

November 12-13*

■ *3d International Workshop on Network and Operating System Support for Digital Audio and Video*

La Jolla, Calif. Sponsor: SIGCOMM, SIGOPS, SIGOIS, UCSD. Contact Venkat Rangan, UCSD, Mail Code 0114, La Jolla, CA 92093-0114; (619) 534-5419; email: venkat@cs.ucsd.edu.

November 16-20

■ *Tri-Ada '92*

Orlando, Fla. Sponsor: SIGAda. Contact Geoffrey O. Mendal, Systems Eng. Research Corp., 415 2348 Leghorn St., Suite 202B, Mountain View, CA 94043; (415) 962-8092; email: mendal@anna.stanford.edu.

December 7-11*

5th International Symposium on Artificial Intelligence

Cancun, Mexico. Sponsor: ITSEM in support by IJCAII and in coop.w/AAAI, CSCSI, IAKE, ECCAI, SMIA and IBM of Mexico. Contact Hugo Terashima, ITSEM, Centro de Inteligencia Artificial, Sucursal de Correos "J", Monterrey, N.L. 64849, Mexico. (52-83) 58-2000 x.5143; fax: (52-83) 58-1400; email: isai@tecmtyvm.bitnet.

December 9-11

■ *5th ACM SIGSOFT Symposium on Software Development Environments*

Washington, DC. Sponsor: SIGSOFT. Contact Ian Thomas, Software Design and Analysis, 444 Castro St., Ste. 400, Mountain View, CA 94041; (415) 694-1464; email: thomas@sda.com.

December 12-16

■ *1992 Winter Simulation Conference*

Arlington, Va. Sponsor: SIGSIM, ASA SCS, ORSE, NIST, TIMS, IIE, IEEE-CS, IEEE-SMCS. Contact Herb Schwetman, MCC, 3500 West Balcones Center Dr., Austin, TX 78759-6509; (512) 338-3428.

December 13-16

ICSC '92: International Computer Science Conference 1992

Hong Kong. Sponsor: IEEE Hong Kong Section, Computer Chapter. Contact Ernest Lam, Dept. of Comp. Sci., Hong Kong Baptist College, Hong Kong; (852) 339-7081; fax: (852) 338-8014; email: ernest@bc750.hkbc.hk.

December 16-18

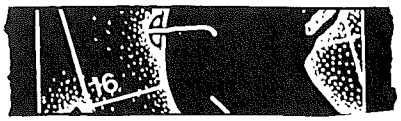
■ *3d Annual International Symposium on Algorithms and Computation*

Nagoya, Japan. Sponsor: ISAAC in coop. w/SIGACT. Contact Osamu Watanabe, Tokyo Inst. of Technology, Dept. of Comp. Sci., Meguro-ku, Ookayama Tokyo 152; email: watanabe@cs.titech.ac.jp.

December 13-16*

Conference on High Performing Networking

Liege, Belgium. For sponsorship and additional information contact Andre Danthine, Univ. de Liege, Institut d'Electricite Montefiore, B 4000 Liege, Belgium; +32 41 56 26 91; fax: +32 41 56 29 89; email: danthine@vm1.ulg.ac.be.



1993

January 3-6

4th International Workshop on Artificial Intelligence and Statistics

Fort Lauderdale, Fla. Sponsor: Society for AI and Statistics and the International Association for Statistical Computing. Contact R.W. Olford and/or P. Cheeseman (519) 888-4609; email: sais@stat.waterloo.edu.

January 4-7

■ *International Workshop on Intelligent User Interfaces*

Orlando, Fla. Sponsor: SIGCHI, SIGART in coop. w/AAAI and BCS and HCI. Contact William Hefley, Software Engineering Inst., SEI 2218, Carnegie-Mellon Univ., 4500 Fifth Ave., (412) 268-7793; email: weh@sei.cmu.edu.

January 17-22

1993 IEEE International Symposium on Information Theory

San Antonio, Tex. Sponsor: IEEE and Information Theory Society. Contact Robert Gray, Electrical Engineering Dept., 133 Durand, Stanford Univ., Stanford, CA 94305; (415) 723-4001; fax: (415) 723-8473; email: gray@isl.stanford.edu.

January 20-22

Conference on Architectures and Compilation Techniques for Fine and Medium Grain Parallelism

Orlando, Fla. Sponsor: IFIP WG 10.3. Contact Kemal Ebcioglu, IBM Thomas J Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598; (914) 945-3267; email: kemal@watson.ibm.com.

February 16-18

■ *CSC '93: 1993 ACM Computer Science Conference*

Indianapolis, IN. Sponsor: ACM. Contact John F. Buck, Indiana Univ., Comp. Sci. Dept., 215 Lindley Hall, Bloomington, IN 47405-4101; (812) 855-7979; email: jbuck@cs.indiana.edu or jbuck@cs.iucs.bitnet.

February 18-19

■ *1993 ACM SIGCSE Technical Symposium*

Indianapolis, In. Sponsor: SIGCSE. Contact Bruce Klein, Dept. of Math and Comp. Sci., Grand Valley State Univ., Allendale, MI 49401-9403; (616) 895-2048; email: 21874bjk@msu.bitnet.

March 17-19

■ *ACM SIGUCCS Computer Center Management Symposium*

St. Louis, Mo. Sponsor: SIGUCCS. Contact Larry Westermeyer, Univ. of Missouri, 8001 Natural Bridge Rd., St. Louis, MO 63121-4499; (314) 553-6000; email: slwwest@umslvma.

April 6-8*

DASFAA-93, 3d International Symposium on Database Systems for Advanced Applications

Deajon, Korea. Sponsor: Korean Information Science Society and Information Processing Society of Japan. Contact Hyoung-Joo Kim, Dept. of Eng., Seoul National Univ., Shinlim-dong, Kwanak-ku, Seoul, Korea; +82 2 880 5327; fax: +82 2 897 0130; email: hjk@krsnucc1.bitnet.

April 18-23

3d International Symposium on Integrated Network Management

San Francisco, Calif. Sponsor: IFIP in coop. w/IEEE CNOM. Contact Action Motivation, P.O. Box 191885, San Francisco, CA 94119; (415) 512-1316; fax: (415) 512-1325; email: 4367585@mcimail.com.

April 19-23

9th International Conference on Data Engineering

Vienna, Austria. Sponsor: IEEE Computer Society Technical Committee on Data Engineering. Contact Forouzan Golshani, Comp. Sci. and Eng. Dept., Arizona State Univ., Tempe, AZ 85287-5406; (602) 965-2855; email: golshani@asuvax.eas.asu.edu.

April 20-23

■ *History of Programming Languages*

Boston, Mass. Sponsor: SIGPLAN. Contact Jan Lee, CIT ITT 133 McBryde Hall, Blacksburg, VA 24061-0119; (703) 231-5780; email: janlee@vtmi.bitnet.

May 14-22

■ *Federated Computing Research Conference*

San Diego, Calif. Sponsor: ACM, CRA, and IEEE. Contact David S. Wise, Indiana Univ., Comp. Sci. Dept., 215 Lindley Hall, Bloomington, IN 47405-4101; email: dswise@cs.indiana.edu.

May 16-18

■ *STOC'93: 25th Annual ACM Symposium on the Theory of Computing 1993*

San Diego, Calif. Sponsor: SIGACT. Contact David S. Johnson, AT&T Bell Labs, 600 Mountain Ave., Rm. 2D-150, Murray Hill, NJ 07974; (908) 582-4742; email: csnet: dsj@research.att.com.

May 17-18*

■ *3d Workshop on Parallel and Distributed Debugging*

San Diego, Calif. Sponsor: SIGPLAN, SIGOPS, incoop.w/ONR. Contact Joan Francioni, Univ. of SW Louisiana, Dept. of Comp. Sci. Lafayette, LA 70504; (318) 231-6602; email: jf@cacs.usl.edu.

May 17-21*

■ *SIGMETRICS 93: Conference on Measurement and Modeling of Computer Systems*

Santa Clara, Calif. Sponsor: SIGMETRICS. Contact Susan Owicki, Digital Equipment Corp., Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301; (415) 853-2270; email: owicki@src.dec.com.

May 19-21*

■ *ACM Symposium in Solid Modeling Foundations and CAD/CAM Applications*

Montreal, Canada. Sponsor: SIGGRAPH. Contact Joshua Turner, Rensselaer Polytechnic Inst., CII-7015, Troy, NY 12180; (518) 276-8713; email: jturner@rdrc.rpi.edu.

CALLS FOR PAPERS

July 1*
9th International Conference on Data and Engineering
Vienna, Austria. April 19-23, 1993. Submit five copies of complete paper to Erich J. Neuhold, GMD-IPSI, Dolivostrasse 15, D-6100 Darmstadt, Germany; (+49) 6151 869 803; email: darmstadt.gmd.de.

July 1
3d International Symposium on Integrated Network Management
San Francisco, Calif. April 18-23, 1993. Submit seven copies of a paper to Yechiam Yemini, Columbia Univ., 450 Comp. Sci. Bldg., New York, NY 10027; or Heinz-Gerd Hegering, Univ. of Munich, Inst. fuer Informatik,-LRZ, Barer Strass 21, D-8000 Muenchen 2, Germany.

July 13
30th Annual Allerton Conference on Communication, Control, and Computing
Monticello, Ill. September 30-October 2. Submit a title and 5-10 page extended abstract to Allerton Conference, Univ. of Illinois, coordinated Science Lab., 1101 W. Springfield Ave., Urbana, IL 61801.

July 15
ICS '92: International Computer Symposium
Taiwan, ROC. December 13-15. Submit four copies (20 double spaced pages max.) and abstract (in English) to An-Chi Liu (Far East region) Information Engineering Dept., Feng Chia U., Taichung, Taiwan, ROC; (886) 4 252 2250 x.3700; email: fcut300@twnmoel 10.bitnet or Ben W. Wah, (USA and other regions), Coordinated Science Labs, Univ. of Illinois, Urbana, IL 61801; (217) 244-7175; email: wah@manip.cs1.uiuc.edu.

August 1
7th Conference on the Scientific Use of Statistical Software (SoftStat '93)
Heidelberg, Germany. March 14-18. Submit all abstracts to SoftStat, ZUMA, Postfach 12 21 55, D-6800, Mannheim I, Germany.

August 3
CSC '93: 1993 ACM Computer Science Conference
Indianapolis, Ind. February 16-18, 1993. Submit five copies of full paper (20 double spaced pages max.) to Stan C. Kwasny, CSC '93 Prog. Chair, Dept. of Comp. Sci., Washington Univ., Campus Box 1045, 1 Brookings Dr., St. Louis, MO 63130-4899; (314) 935-6123; email: sck@cs.wustl.edu.

August 14*
Working Conference on Architectures and Compilation Techniques for Fine and Medium Grain Parallelism
Orlando, Fla. January 20-23, 1993. Submit four copies of paper (20 pages max.) to Jean-Luc Gaudiot, Univ. of Southern California, Dept. of Electrical Engineering-Systems, Los Angeles, CA 90089-2563; (213) 740-4484; fax: (213) 740-4449; email: gaudiot@usc.edu.

September 12*
DASFAA-93, 3d International Symposium on Datrabase Systems for Advanced Applications
Deajon, Korea, April 6-8, 1993. Submit six copies of full paper (5,000 words max. in English) to SongChun Moon, KAIST, 207-43 Cheongrkyang Dongdaemun, Seoul 130-012, Korea; fax: +82-2-960-6743; email: moon @dbsun2.kaist.ac.kr.

October 15*
Solid Modeling '93: 2d Symposium on Solid Modeling and Applications
Montreal, Canada. May 19-21, 1993. Submit one full paper (5,000 words max.) to Mary Johnson, CII 7015; Rensselaer Polytechnic Institute; Troy, NY 12180; (518) 276-6751; (518) 276-2702; email: mjohnson@ rdrc.rpi.edu.

October 15
TapSoft '93
Orsay, France. April 13-17, 1993. Submit five copies of original papers to J.P. Jouannaud, TapSoft '93, AFCET, 156 Boulevard Perieire, 75017 Paris, France; 33 1 47 66 24 19; fax: 33 1 42 67 93 12.

October 31*
Graphics Interface '93
Toronto, Ontario, Canada. Submit five copies of a full paper (20 double-spaced max.) with full names, addresses for all authors to Tom Calvert, GI '93, School of Computing Science, Simon Fraser Univ., Burnaby, British Columbia V5A 1S6, Canada; (604) 291-3610; fax: (604) 291-3045; email: tom@sfu.ca.

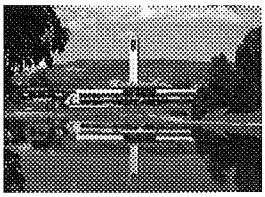
November 1*
7th International Symposium on Methodologies for Intelligent Systems
Trondheim, Norway. June 15-18, 1993. Submit four copies complete (20 pages max) to Jan Kmorwski, Univ. of Trondheim, Norwegian Institute of Technology, Dept. EE and Comp. Sci., N-7034 Trondheim, Norway; email: janko@idt. unit.no. or Zbigniew W. Ras, UNC-Charlotte, Dept. of Comp. Sci., Charlotte, NC 28223.

November 2*
2d International Workshop Logic Programming and Non-Monotonic Reasoning
Lisbon, Portugal. June 28-30, 1993. Submit six copies (double-spaced, 20 pages max.) LP &NMR-93, Attn: Anil Nerode, Mathematical Sciences Institute, Cornell Univ., Ithaca, NY 14853.

TOOLS USA '92

Technology of Object-Oriented Languages and Systems

Program Chair: Raimund Ege • Conference Chair: Bertrand Meyer • Panel Chair: Madhu Singh



EIGHTH INTERNATIONAL CONFERENCE AND EXHIBITION
University of California, Santa Barbara, August 3-6, 1992

THE PREMIER APPLICATION-ORIENTED EVENT ENTIRELY DEVOTED TO OBJECT-ORIENTED TECHNOLOGY

TOOLS USA '92 LINKS INDUSTRY WITH RESEARCH

TUTORIALS:
August 3-4

Teaching O-O Programming & Design: James McKim (USA)
O-O Analysis & Design: Kim Walden (Sweden)
Frameworks: A Programmer's View of Developing Reusable Software: Ralph Johnson (USA)
O-O Specification for Real-Time Systems: Dino Mandrioli (Italy)
O-O Programming in Modula-3: Samuel Harbison (USA)
Constraint-Based Languages and Systems: Alan Borning (USA) and Bjorn Freeman-Benson (Canada)
O-O Concurrent Programming: Klaus-Peter Löhr (Germany)
Developing Scientific Software with Eiffel: David Butler (USA)
Applying O-O Technology in the MIS World: Jean-Marc Nerson (France)
O-O Programming Languages: Raimund Ege (USA)
Testing O-O Software: Ed Berard (USA)
O-O Databases: Won Kim (USA)
O-O User Interfaces: Arno Gourdol (France)
O-O Methods and Ada: Henry Baker (USA)
Theoretical Basis of O-O Programming: John Mitchell (USA)
O-O Technology–The Managerial Perspective: Bertrand Meyer (USA)

CONFERENCE SESSIONS:
August 5-6

- Application and Experience
- Modeling
- Database Access
- Concurrency
- Theoretical Issues

WORKSHOPS:

- User Interface Design
- O-O Methods for Scientific Computation
- O-O Analysis and Design
- O-O Databases
- O-O in the MIS World
- Typing

Keynote: Peter Deutsch, Sun Microsystems:
Objects: Challenges Beyond Languages and Applications

TOOLS provides a unique opportunity to meet industry practitioners of object-oriented technology, and hear from the best international experts about the latest developments in the field.

REGISTER EARLY
AND GET A DISCOUNT

TOOLS USA '92
P.O. Box 6863
Santa Barbara, CA 93160
Tel: (805) 685-1006
Fax: (805) 685-6869
E-mail: tools@eiffel.com

Please send me the complete program of the conference by: ❑ Mail ❑ Fax ❑ E-mail

Last Name _____ First _____

Company _____ Job Title _____

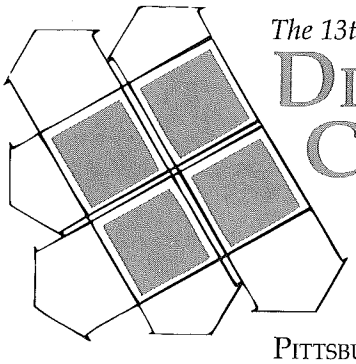
Company Address _____

City _____ State/Zip _____ Country _____

Phone _____ Fax _____

E-Mail _____ ❑ **My company is interested in exhibiting. Please send me an exhibitor information kit.**

CALL FOR PAPERS



The 13th International Conference on

DISTRIBUTED COMPUTING SYSTEMS

PITTSBURGH HILTON
PITTSBURGH, PENNSYLVANIA, USA
MAY 25-28, 1993

This conference encompasses the technical aspects of specifying, designing, implementing, and evaluating distributed computing systems. In such systems, there are multiple processing resources interconnected to cooperate under system-wide control with minimal reliance on centralized procedures, data, or hardware. The location of computing resources may span the spectrum from physical adjacency to geographical dispersion. The topics of interest include the following aspects of distributed computing systems.

- Computer Architecture and Distributed Shared Memory
- Cooperative Work and Artificial Intelligence
- Languages, Tools, and Software Engineering
- Distributed System Services and Management
- Distributed Operating Systems
- Communication Architectures and Protocols
- Multimedia Computing and Communication
- Modeling and Performance Evaluation
- Reliability and Fault Tolerance
- Distributed Algorithms
- Distributed Databases
- Real-Time Issues

INFORMATION FOR AUTHORS

Authors are requested to submit six copies (in English) of their double-spaced typed manuscript (maximum of 20 pages) with an abstract and keywords to Prof. Wittie by *Thursday, October 15, 1992*. The conference language is English and final papers are restricted to eight IEEE model pages. Each paper must be accompanied by a submission letter that indicates which one or two conference areas are most relevant. If there are multiple authors, one author must be designated as responsible for correspondence and preparation of the camera-ready paper for inclusion in the proceedings. Please give postal address, email address, and telephone for the corresponding author. Authors will be notified of acceptance by February 1, 1993 and will be given instructions for final preparation of their papers at that time.

Submit papers to:
Larry Wittie
Z4400 Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794-4400, USA
Tel: (516) 632-8456
Fax: (516) 632-8334
E-mail: lw@sbcs.sunysb.edu

TUTORIALS

In addition to papers, proposals for one day tutorials are solicited in any of the conference areas. Proposals should be submitted to Dr. Yao-Nan Lien by *October 1, 1992*.

Submit tutorial proposals to:
Yao-Nan Lien
AT&T Bell Laboratories
200 Park Plaza, Room IHP 2A340
Naperville, IL 60566-7050, USA
Tel: (708) 713-4318
Fax: (708) 713-7098
E-mail: yaonan.lien@att.com

FOR MORE INFORMATION, PLEASE CONTACT:

Benjamin W. (Ben) Wah
Coordinated Science Laboratory
University of Illinois, MC228
1101 W. Springfield Avenue
Urbana, IL 61801-3082
Tel: (217) 333-3516
Fax: (217) 244-7175
E-mail: b-wah@uiuc.edu

SPONSORED BY:

IEEE Computer Society

THE INSTITUTE OF ELECTRICAL & ELECTRONICS ENGINEERS, INC.
IEEE

ORGANIZING AND PROGRAM COMMITTEES

General Chair
Benjamin W. Wah, University of Illinois at Urbana, USA

Program Chair
Larry Wittie,
SUNY at Stony Brook, USA

Computer Architecture and Distributed Shared Memory Vice Chair
Michel Dubois, University of Southern California, USA

Cooperative Work and Artificial Intelligence Vice Chair
Nick J. Cercone, Simon Fraser University, Canada

Languages, Tools, and Software Engineering Vice Chair
Gail E. Kaiser, Columbia University, USA

Distributed System Services and Management Vice Chair
Liba Svobodova, IBM Zurich Research Laboratory, Switzerland

Distributed Operating Systems Vice Chair
Partha Dasgupta, Arizona State University, USA

Communication Architectures and Protocols Vice Chair
Haruhisa Ichikawa, NTT Software Research Laboratory, Japan

Multimedia Computing and Communication Vice Chair
Ralf Steinmetz, IBM European Networking Center, Germany

Modeling and Performance Evaluation Vice Chair
Alexander Thomasian, IBM Watson Research Center, USA

Reliability and Fault Tolerance Vice Chair
Hermann Kopetz, Technical University of Vienna, Austria

Distributed Algorithms Vice Chair
Divyakant Agrawal, University of California at Santa Barbara, USA

Distributed Databases Vice Chair
Wojciech Cellary, French-Polish School of New Information and Communication Technologies, Poland

Real-Time Issues Vice Chair
Hideyuki Tokuda, Carnegie Mellon University, USA

Tutorials Chair
Yao-Nan Lien, AT&T Bell Laboratories, USA

Awards Chair
Joseph E. Urban, Arizona State University, USA

Publicity Chair
Bruce McMillin, University of Missouri-Rolla, USA

International Liaison Chairs
Makoto Takizawa, Tokyo Denki University, Japan
A. M. Tjoa, University of Vienna, Austria

Local Arrangements Chair
Mario R. Barbacci, Carnegie-Mellon University, USA

Treasurer
Susan D. Urban, Arizona State University, USA

TC on Distributed Processing Chair
Bill Buckles, Tulane University, USA

Steering Committee Chair
Mike Liu, Ohio State University, USA

The Twenty-Second Annual International Symposium on Fault-Tolerant Computing

FTCS 22

Sponsored by IEEE Computer Society, The University of Massachusetts, Texas A&M University and the Charles Stark Draper Laboratory, Inc. in co-operation with IEEE Technical Committee on Fault-Tolerant Computing and IFIP Working Group 10.4

The Lafayette Hotel **Boston, Massachusetts USA** **July 8 - 10, 1992**

****

SCOPE

The Technical Program will cover the following topics:

- Architecture
- Communication Protocols
- Modeling and Simulation
- Error Correcting Codes
- Synthesis for Testability
- System Fault Detection and Reconfiguration
- Self-Checking and Diagnosis
- Fault Tolerant Hypercubes and Meshes
- Scheduling and Fault Classification
- Wafer Testing and Defect Toleranc
- Fault Injection
- Test Pattern Generation
- Recovery
- Fault Tolerance Theory
- Field Experience

In conjunction with FTCS-22, a **workshop on Fault-Tolerant Parallel and Distributed Systems**, will be held in Amherst, MA, July 6-7. For further information call Don Fussell at (512) 471-9719 or send e-mail to fussell@cs.utexas.edu. If you cannot attend the workshop, you may order the *Digest*; see below.

EXHIBITS For the first time, FTCS will be providing a comprehensive program of commercial and university exhibits. The following have planned to exhibit: *Charles Stark Draper Laboratory, Digital Equipment Corporation, IBM, Integrated Micro Products, Sequoia Systems, Stratus Computers, Tandem Computers, MIT Press, Digital Press, Carnegie-Mellon University, and Texas A&M University.* In addition to their exhibits, the exhibitors will be presenting informative technical briefings during the "Exhibitors' Forum" all day on Thursday, July 9. A special exhibits-only registration fee will be available. Call:(409) 862-2438. For other information, contact Professor Dhiraj K. Pradhan, Symposium Chair, at (409) 862-2438.

ABOUT BOSTON The conference site is The Lafayette Hotel, located within convenient walking distance of major attractions such as the Freedom Trail, the Boston waterfront, the State House and Faneuil Hall. Short trips to such spots as Beacon Hill are within easy reach where a walk along gas-lit brick sidewalks and cobblestone streets is reminiscent of the nineteenth century. SAIL BOSTON 1992 (Tall Ships) are expected. For other Boston information, contact the Boston Tourist Bureau, (617) 536-9427.

EVENTS On Wednesday afternoon, July 8, we will have a scenic two-hour cruise in Boston Harbor including a preview of SAIL BOSTON 1992, with music by the "New Black Eagle Jazz Band." The cruise will be followed by a buffet dinner and private admission to Boston's Computer Museum. The banquet on Thursday evening features a sumptuous reception and dinner, followed by a technical talk by a leading commercial developer of fault-tolerant systems. Guest tickets will be available for all these events.

HOTEL INFORMATION: The Lafayette Hotel: One Ave. de Lafayette; Boston, MA 02111 Tel: (800) 621-9200
Single/double @$99.00 + 9.7% MA tax.

FTCS-22 REGISTRATION INFORMATION

NAME:_____
CO._____mail stop_____
ADDRESS_____
CITY_____STATE_____ZIP_____
COUNTRY_____ TEL:_____
IEEE/Computer Society Member No._____

Send Registration To:
Marsha Lee
Computer Science Dept.
Texas A&M University
H. R. Bright Building
College Station, TX 77843
Tel: (409)862-2438
Fax: (409) 847-8578

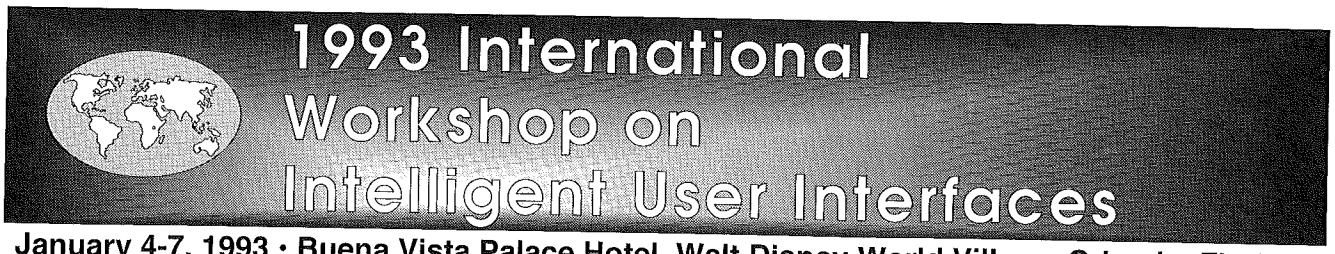
Registration Fees	IEEE Member	Non-Member	Student
	By 6/15 US $350_____	US $425 _____	US $90_____
	After 6/15 US $400 _____	US $495 _____	US $110_____

Registration includes: FTCS-22 *Digest*, lunch-7/8, receptions, (cruise, museum, dinner), and banquet.

Additional copies of FTCS-22 *Digest*: US $40.00 _____ **Workshop *Digest*** US $25.00_____
COMPANIONS: Lunch, 7/8 $25.00_____ Cruise, museum, dinner $45.00_____
Banquet $55.00_____Special Dietary Requirements:_____**TOTAL FEES $**_____

SEND FULL PAYMENT IN US$ WITH THIS FORM. USE CHECK OR CREDIT CARD. PURCHASE ORDERS ARE **NOT** ACCEPTED. MAKE CHECKS PAYABLE TO **FTCS-22** . USE YOUR CREDIT CARD TO REGISTER BY FAX. WE WILL CONFIRM REGISTRATION BY MAIL.

FOR CREDIT CARD PAYMENT. Check card type: VISA____ MASTERCARD_____ AMEREXPRESS_____
CARD NO.:_____EXPIR. DATE:_____CARDHOLDER SIGNATURE_____



January 4-7, 1993 • Buena Vista Palace Hotel, Walt Disney World Village, Orlando, Florida

Call for Participation

General Co-Chairs:

William E. Hefley
Carnegie Mellon University
U.S.A.

Dianne Murray
University of Surrey, U.K.

Treasurer:

Steven Roth
Carnegie Mellon University
U.S.A.

Publications:

Wayne Gray
Fordham University, U.S.A.

Program Committee:

David Benyon
The Open University, U.K.

Joelle Coutaz
IMAG, Grenoble, France

Steven Feiner
Columbia University, U.S.A.

Saul Greenberg
University of Calgary,
Canada

Bob Kass
EDS Center for
Advanced Research, U.S.A.

Johanna Moore
University of Pittsburgh,
U.S.A.

Lisa Neal
EDS Center for
Advanced Research, U.S.A.

Reinhard Opperman
GMD, Germany

Elaine Rich
MCC, U.S.A.

Michael Wilson
Rutherford Appleton
Laboratory, U.K.

Mailing Address:

c/o Bill Hefley
Software Engineering
Institute
Carnegie Mellon University
Pittsburgh, PA 15213
U.S.A.
+1-412-268-7793
ii-Workshop93.chi@
xerox.com

Advances in multidisciplinary research are creating a new generation of intelligent interfaces that use computers to facilitate the interaction of people with information, knowledge, tasks and situations. These interfaces rely on advanced computational techniques and cognitive science theories to build systems that enhance user performance. Such systems can help users to accomplish complex tasks by interpreting ambiguous input, phrasing multimodal output in ways sensitive to users' abilities and situations, and providing effective advice and assistance.

This new event stems from a prior successful workshop in 1988 on Architectures for Intelligent Interfaces (sponsored by AAAI/ACM SIGCHI). The 1993 International Workshop will focus on a diversity of approaches to human-computer interaction, from those employing advanced computational techniques to those incorporating cognitive and user models to amplify human cognitive abilities.

The goal of the workshop is to explore ways in which techniques for knowledge representation, inferencing, modeling, and presentation can provide the adaptability and reasoning capabilities required for more intelligent human-computer interaction. We aim to stimulate high quality discussion amongst participants from different countries and from disciplines such as cognitive science, human-computer interaction, computational linguistics and artificial intelligence. The workshop will bring together researchers and practitioners with an interest in methods, techniques, tools, and technology for constructing and evaluating intelligent systems.

The intimate size, single track, and comfortable surroundings make this workshop an ideal opportunity to exchange research results and implementation experiences. Attendance will be limited to 100 participants with presentations spread over three days. The format is split into two and a half days of presentations with one half day of working sessions, enhanced by three thought-provoking plenary speakers.

Interested participants should submit an original paper for review. Submissions addressing theory, system building, or evaluation issues are welcomed. Papers may address human-computer interaction or artificial intelligence/computational perspectives on (but not limited to) the following topics:

- Intelligent User Interfaces in a diverse range of application areas including tutoring and advisory systems; natural language processing; generation and understanding of nonverbal media; planning and explanation; information retrieval; computer-supported cooperative work, decision support and supervisory control.
- Interface-Building Tools and Techniques: knowledge-based and user modelling techniques for intelligent interface design, including plan and intent recognition, automatic presentation, explanation, user aiding (aids, critics, tutors), knowledge representation and modeling of users, systems, tasks.
- Intelligent front-ends to interactive, multimedia, hypermedia, and knowledge-based systems.
- Adaptive and customisable systems.
- Intelligent Agents and agent-based interaction.
- Requirements and architectures for intelligent, cooperative, and multimodal interfaces.
- Methods for analyzing, designing & evaluating users' needs & performance, intelligent interfaces & systems.

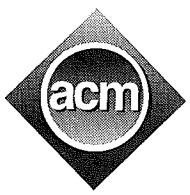
SUBMISSIONS

Papers will be accepted either for presentation as talks or posters. Authors are encouraged to submit work-in-progress to the poster session whilst longer papers should highlight both the general scientific contribution of the research and its practical significance. Six camera-ready copies of original papers written in English should be submitted, as long papers (presented as a talk: 8 pages) or as short papers (interactive poster presentation: 4 pages). Submitted papers must be unpublished, substantively different from papers currently under review and must not be submitted elsewhere before notification date. A book, including revised papers and workshop results, may be published following the meeting.

Each paper should have a separate cover page containing the title of the paper; author(s) and affiliation(s), contact address of main author, phone, fax and e-mail address; an abstract (100-200 words) and a list of up to five key words or phrases linked to the submission topics describing its content. Detailed formatting instructions are available (for an example, see the SIGCHI Bulletin, July 1991, pp. 93-96).

DEADLINES

You are requested to send notification of intention to participate and potential titles of submissions before July 1, 1992, to ii-Workshop93.chi@xerox.com. Interested participants should forward SIX camera-ready copies of their paper to Wayne Gray, Graduate School of Education, Room 1008, Fordham University, 113 West 60th Street, New York, NY 10023 U.S.A.; +1-212-636-6464. E-mail or fax submissions will not be considered. Papers must be received before July 17, 1992. Authors will be notified by September 25, 1992. For further information, contact the Conference Secretariat at ii-Workshop93.chi@xerox.com or at the mailing address on the left.



Sponsored by ACM SIGCHI - Special Interest Group on Computer & Human Interaction and
SIGART - Special Interest Group on Artificial Intelligence
In Cooperation with AAAI (American Association for Artificial Intelligence) & the British HCI Group

INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING

IFIP CONGRESS '92

FROM RESEARCH TO PRACTICE

12th WORLD COMPUTER CONGRESS MADRID-SPAIN September 7-11, 1992

STREAM (5 DAYS): SOFTWARE DEVELOPMENT AND MAINTENANCE

AREAS OF INTEREST:
- Programming environments and tools (design, implementation, and application)
- Formal methods
- Program and system reusability
- Software development methodologies, practice, and experience
- Programming language design and usage
- Technology transfer

INVITED SPEAKERS: Prominent figures will be invited to talk on Programming Environments, Formal Methods, Software Development, and Programming Languages

STREAM (5 DAYS): ALGORITHMS AND EFFICIENT COMPUTATION

AREAS OF INTEREST:
- Data structures and graph algorithms
- On-line and continuous algorithms
- Computational geometry, computer graphics and robotics
- Combinatorial optimization
- Parallel and distributed algorithms
- Novel computational models
- Computational learning
- Complexity theory and cryptography
- Symbolic computation

INVITED SPEAKERS: Prominent figures will be invited

STREAM (5 DAYS): FROM ARCHITECTURES TO CHIPS

AREAS OF INTEREST:
- Parallel and distributed computing
- Supercomputers
- Hardware/software dependability
- Formal aspects of hardware design
- VLSI design, design tools, CHDLs, frameworks
- Workstation concepts and graphics
- Neural networks
- Performance & reliability modeling & evaluation.

INVITED SPEAKERS: Hajime Ishikawa, Jean-Claude Laprie, Alain Martin, David May, John Meyer, Michele Morganti

STREAM (5 DAYS): INFORMATICS AND EDUCATION

AREAS OF INTEREST:
- The changing role of university computing centers
- Computer equity
- Applying research to support learners
- Teleteaching

INVITED SPEAKERS: Magda Bruin, Geoff Cumming, Stephen Franklin, Robert Lewis

STREAM (5 DAYS): THE VULNERABILITY OF THE INFORMATION SOCIETY: SOCIAL, LEGAL, AND SECURITY ASPECTS

AREAS OF INTEREST:
- Opportunities and risks in the adoption of Information Technology.
- Legal aspects
- Reliability and security in PCs and LANs
- Impact of the vulnerability in the workplace and on the general public
- Identification and authentication
- The Electronic Cottage: Information and communication technologies at home
- Computer ethics and professional responsibility
- International Development esp. in Latin America

INVITED SPEAKERS: G.B.F. Niblett, Herbert Kubicek, Lance Hoffman, Harold W. Highland

SUBCONFERENCE (2 1/2 DAYS): EXPANDING THE POWER OF THE PERSONAL COMPUTER

AREAS OF INTEREST:
- Man-machine interfaces
- End-user programming
- Cooperative work group interaction
- PC-LAN and WAN networking
- Paperless office
- Multilingual support
- Integration of multimedia
- Database and document technologies
- Computer aided engineering

INVITED SPEAKERS: Esther Dyson, Dave Liddle, Stan Rosenschein, Rob Shostak, Jeff Raikes, Rolf Wildhack, Peter Maier

SUBCONFERENCE (2 1/2 DAYS): ENHANCING THE INTELLIGENCE IN INFORMATION SYSTEMS

AREAS OF INTEREST:
- Design criteria for incorporating intelligence and learning in systems
- Acquisition of knowledge and its validation, verification, and testing
- Application of knowledge-based systems to analysis, design, and maintenance
- Implementation problems and their solutions
- Culture and stage of development in intelligent systems
- Frontiers in research

INVITED SPEAKERS: Prominent figures will be invited

HONOUR COMMITTEE: Presided over by their Majesties the King and Queen.
ORGANIZING COMMITTEE CHAIRWOMAN: Rosa Alonso
TECHNICAL COMMITTEE CHAIRMAN: Wilfried Brauer
KEYNOTE SPEAKERS: Michel Carpentier, Josep Cornu, Josep M. Vilá, Philip Kahn, Pamela McCorduck.

TUTORIALS:

TUTORIAL 1: Computer security for small systems. Dr. Harold J. Highland
TUTORIAL 2: Academic computing: supporting an integrated heterogeneous environment. Stephen D. Franklin; Bernard Levrat.
TUTORIAL 3: Information systems methodologies. A framework for understanding. T. W(Will) Olle; A. A(Alex) Verrjsn Stuart
TUTORIAL 4: Automated approaches to formal hardware analysis and verification. Randal E. Bryant.
TUTORIAL 5: Privacy and freedom: can we have them all in the next century?. Lance J. Hoffman.
TUTORIAL 6: Validation and knowledge-based systems: consistency checking and testing. Jean Pierre Laurent.

WORKSHOPS:

WORKSHOP 1: Cooperative research programs. Brian Oakley.
WORKSHOP 2: Informatics for environmental protection. B. Page.
WORKSHOP 3: Encouraging academic/practitioner collaboration. Bernard Glasson.
WORKSHOP 4: Continuous algorithms and complexity. Josep Traub.

TECHNICAL VISITS:

R&D Alcatel; R&D Telefónica; ATT Microelectronics; IBM; Computer Science School; Technological Park; Telecommunications Engineering School.

POSTER SESSIONS

CONGRESS REGISTRATION (in pesetas)

	Through May 4	May 5 July 7	July 16 Through on-site
DELEGATE FEE	44.000	52.000	59.000
STUDENTS WITH PROCEEDINGS	16.000	21.000	26.000
STUDENTS WITHOUT PROCEEDING	11.000	16.000	21.000
ACCOMPANYING PERSON.			
Number of persons x	1.000	2.000	3.000
TUTORIAL (Each)	17.000	22.000	26.000
TECHNICAL VISITS (Each)	2.000	2.500	3.000

HOST: FESI (Federación Española de Sociedades de Informática)

FOR ADVANCE PROGRAM AND REGISTRATION FORM, CORRESPOND WITH:

GRUPO GEYSECO C/ Mauricio Legendre, 4-8ºG
28046 MADRID (SPAIN)

CALL FOR PAPERS

Twelfth Annual IEEE

INTERNATIONAL Phoenix CONFERENCE on COMPUTERS and COMMUNICATIONS

March 24–26, 1993

Scottsdale, Arizona

Sponsored by:
INSTITUTE OF ELECTRICAL AND ELECTRONIC ENGINEERS

and

IEEE COMMUNICATIONS SOCIETY

In cooperation with:
IEEE COMPUTER SOCIETY,
UNIVERSITY OF ARIZONA,
and
ARIZONA STATE UNIVERSITY

CONFERENCE CHAIR
James Weeldreyer
Honeywell, Inc.
16404 N. Black Canyon Highway
Phoenix, AZ 85023 USA
Tel: 602-436-4813
FAX: 602-436-4848
E-Mail: Weeldrey@iasdv1.iasd.honeywell.com

IMPORTANT DATES:

Submission Deadlines.
For papers, special sessions, and tutorials:
17 July 1992
For Technical Demonstrations:
28 August 1992.

Acceptance notifications.
For papers:
25 September 1992
For special sessions (provisional) and tutorials:
21 August 1992
technical demonstrations
19 October 1992

Camera ready version of papers due 30 November 1992

Conference Dates
24-26 March 1993

This international conference provides a forum for the presentation and exchange of current work in computers, communications, their synergism, and their applications. We particularly solicit industrial, business, and government participation as well as the active involvement of the academic community. We know it is vital that there be a dialogue between practitioners and researchers. Thus, in addition to research contributions, we solicit reports detailing experiments, evaluation, problems, and opportunities associated with design, implementation, and operation.

PAPERS
Submitted manuscripts must be no longer than 5000 words, be typed double-spaced, and include an abstract of approximately 300 words. Longer papers and reports will not be considered. Authors should obtain company and government clearances prior to submission of the papers. Please submit five copies of complete paper and abstract by 17 July 1992 to:

Robert Meltz, Program Chair
Arizona State University
Department of Aeronautical Technology
Tempe, AZ 85287-6406 USA
Telephone: 602-965-7775
FAX: 602-965-5089
E-Mail:idrom @ asuvm. inre.asu.edu.

All papers submitted will be reviewed by the Program Committee. They will be judged with respect to their quality, originality, and relevance. Authors will be notified of acceptance/rejection shortly after 25 September 1992. Accepted papers will be published in the IPCCC Proceedings. Awards will be given for the best paper and for the best presentation. Outstanding papers will be considered for publication in related IEEE journals.

SPECIAL SESSIONS
We solicit proposals for special topics and panel sessions. Please contact the Program Chair for proposal guidelines.

TUTORIALS
Proposals for one-day tutorials related to the conference topics are desired. Please contact the Tutorials Chair for tutorial proposal guidelines. Proposals should be sent by 17 July 1992 to:

Alan Johnsey
Bull HN Information Systems
P.O. Box 8000
Mail Stop B8
Phoenix, AZ 85066 USA
Telephone: 602-862-4036
FAX: 602-862-4750
E-Mail:a.johnsey@az05.bull.com

TECHNICAL DEMONSTRATIONS
We are actively soliciting proposals for Technical demonstrations from industry and academia. These demonstrations may relate to any of the conference topics and may include work in progress, new products or research prototypes. Please contact the Demonstration Chair for proposal guidelines. Proposals should be sent by 29 August 1992 to:

Frank Calliss, Demonstration Chair
Arizona State University
Dept. of Computer Science & Eng.
Tempe, AZ 85287-5406 USA
Telephone: 602-965-2804
FAX: 602-965-2751
E-Mail:callis@asuvax.eas.asu.edu

SUGGESTED TOPICS:

Computer Technology
- Parallel and Distributed Computing
- Fault Tolerance and Reliability
- Neural Network Computing
- Distributed Database Systems
- Optical Disk Storage
- VLSI/VHSIC Developments
- Advanced Architectures

Communications Technology
- Fiber Optics
- Satellite/Terrestrial Systems
- Communications Theory
- Spread Spectrum

Software Systems
- Specification Methodologies
- Development Environments
- Object-Oriented Systems
- Real-Time Systems
- Performance Measurement and Evaluation
- Graphics and Scientific Visualization
- Distributed Operating Systems
- AI/Expert Systems
- Applications

Networking Systems
- Protocols
- Interoperability
- Local and Wide Area Networks
- Frame Relay
- Management, Control and Security
- Performance
- ISDN Systems
- Applications

Strategic Impact
- Transition To Open Systems
- Network Technology Choices
- Alternatives In Software Engineering Technology
- Communications and Information Resources In Business Decisions
- Project Management

CAREER OPPORTUNITIES

University of Oregon

The Department of Computer and Information Science invites applications for a senior faculty position created by a new state Centers of Excellence award. We are seeking a person who will be an active leader in the department, willing to serve a term as department head and also play a key role in relations to the computer industry. Applicants should have a Ph.D. in computer science or related field and a distinguished record of teaching and research in the area of parallel processing (including parallel architectures, languages, and performance modeling) or human-computer interaction (including computer graphics and scientific visualization). Our department has 14 other research faculty positions (including one other new position for which we are currently recruiting), approximately 20 Ph.D. students, 50 M.S. students, and 150 B.S students. We have strong research programs in parallel and distributed systems, computer graphics, user interfaces, programming languages, software engineering, artificial intelligence, and theoretical computer science, and active interdisciplinary ties with other on-campus groups in the fields of cognitive science, neuroscience, economics, biology, physics, and mathematics. We offer a modern computing environment (a MasPar MP-1100, two Sequent Symmetry multiprocessors, and dozens of Sun and HP workstations) housed in a new computer science building.

Review of applications will continue until the position is filled. The position is available September 1992, with a target date for filling the position by January 1993. Qualified applicants should send their curriculum vitae and the names of at least three references to: Professor John Conery, Faculty Search Committee, Department of Computer and Information Science, University of Oregon, Eugene, OR, 97403-1202. For more information send e-mail to conery@cs.uoregon.edu or phone (503)-346-3973. The University of Oregon is

an Equal Opportunity/Affirmative Action Employer committed to cultural diversity. We especially encourage applications from women and minorities.

University of South Florida

The Department of Computer Science and Engineering is seeking applicants for a visiting faculty position at the Sarasota Campus starting Fall Semester, 1992. The Department offers bachelor's degrees in Computer Science, Computer Engineering, and Information Systems, and has Master's and Ph.D. programs in Computer Science and Engineering.

The University of South Florida, with an enrollment of more than thirty-three thousand, is the second largest institution in the State University system. The main campus is located in Tampa, the principal city of dynamic West Central Florida. The Sarasota campus is located sixty-five miles south of Tampa in an area known for its cultural resources. The University operates an interactive instructional television network that links the Tampa, Sarasota, St. Petersburg and Lakeland campuses and a large number of industrial sites.

The Department is headquartered in a new twelve million dollar building which it shares with the Department of Electrical Engineering on the Tampa Campus. All of the present fifteen full-time faculty members are assigned to the Tampa campus. The successful applicant will be assigned to the Sarasota campus and will engage in undergraduate teaching, advising and research. Travel between the Sarasota and Tampa campuses and substantial departmental interaction will be required. The Department research network includes a substantial number of SUN, DEC and IBM workstations, an INTEL 2/386 hypercube, and a variety of specialized graphics and image processing equipment. Additional computing resources are available on the College computing network and the University network. All three networks are accessible on the Sarasota campus. Some specialized equipment is accessible only on the Tampa campus. The departmental faculty are actively engaged in research activities supported by federal, state and industrial sources. There is industrial interaction with a number of companies in West Central Florida including AT&T, E-Systems, GTE Data Services, Group Technologies, Harris, Hercules, Honeywell and IBM.

Applicants should send an updated resume and arrange to have three letters of reference sent to Faculty Search Committee, Computer Science and Engineering, University of South Florida, Tampa, Florida 33620.

The University of South Florida is an equal opportunity and affirmative action employer.

Southern University at Baton Rouge

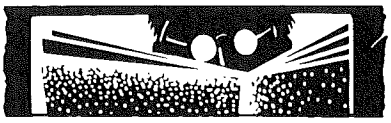
Chairperson
Computer Science Department

The Computer Science Department invites applications and nominations for the position of chairperson. Applications must be postmarked by July 1, 1992, and the anticipated starting date is August, 1992.

Applicants for the position should have the earned doctorate in computer science, an established record of research and teaching at both the graduate and undergraduate levels, and demonstrated leadership ability. This is a full-time, nine-month, tenure track position, with an opportunity to work during summer sessions as desired.

The salary is negotiable, depending on rank, experience and other qualifications.

The Computer Science Department has approximately 400 undergraduate majors and 90 graduate students. Degrees offered include the B.S. degree in Computer Science and Computer Information Systems and the M.S. degree in Information Systems, Operating Systems, Mini/Micro Systems, and Educational Computing. The Science Option of the undergraduate program is accredited by the Computer Science Accrediting Commission of the Computing Sciences Accreditation Board (CSAB). The department has 15 full-time faculty with research interests in networks, database management, software engineering, numerical algorithms and computer science curriculum development.



Computer equipment available in the department includes a VAX 8200, two IBM PC laboratories, and an AT&T 3B2 laboratory. Faculty and students have access to the University's IBM ES/9000 Model 260 and the Computer Science Department and College of Engineering share an IBM 4341.

Southern University at Baton Rouge is an historically Black land-grant college with 9,000 students. The University has nine degree-granting colleges, in addition to the Graduate School. Its campus is situated on bluffs overlooking the majestic Mississippi River in the capitol city of Baton Rouge.

Applications should include a complete resume and three professional references and should be sent to Mrs. Beulah Clark, Chairperson, Computer Science Chair Search Committee, P.O. Box 9221, Southern University, Baton Rouge, LA 70813. For further information, call (504) 771-2060 or Fax (504) 771-4223.

Southern University is an Equal Opportunity Institution.

D. E. Shaw & Co.

Algorithmic Trading

D. E. Shaw & Co. is a small (several dozen employees), highly capitalized (over $300 million in partners' equity), extremely successful Wall Street firm specializing in quantitative finance and computational trading. Computer scientists and system designers form the professional core of the firm, and not merely its support staff, and participate in its profits. Our technical staff includes both Ph.D.-level researchers recruited from Stanford, MIT, and other leading computer science departments and extraordinarily talented B.S.- and M.S.-level system designers and "superhackers". It is our practice to compensate unusually gifted individuals at a level exceeding that of the market. Applicants should send resumes to The Recruiting Department, D. E. Shaw & Co., 39th Floor, Tower 45, 120 W. 45th Street, New York, NY 10036.

WEB DEVELOPMENT CORPORATION

WEB Development Corporation invites applications for the position of:

RESEARCH ASSOCIATE

An ideal candidate should have experience in pattern recognition, signal processing or related fields, as well as extensive programming expertise.

A Ph.D. degree in Mathematics, Computer Science or Electrical Engineering is required.

WEB Development is a small research laboratory near Philadelphia whose current activities span computer science, cognitive science, optics and physical chemistry.

Please send resume with two references to:

Dr. Mahmut Gunar
WEB Development Corporation
Longwood Corporate Center South
415 McFarlan Road
Rennett Square, PA 19348

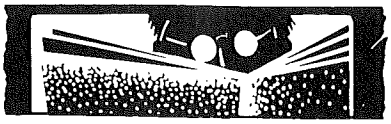
NATIONAL UNIVERSITY OF SINGAPORE

DEPARTMENT OF INFORMATION SYSTEMS AND COMPUTER SCIENCE (DISCS)

Applications are invited for academic positions in the Department. PhD in CS or IS required. Rank and salary will depend on qualifications and experience. Well-qualified applicants in all areas of the subjects will be considered, but research/teaching experience in the following areas are specially sought: systems analysis and design, software engineering, operations research, EDP auditing and financial modelling.

Competitive salary and fringe benefits include: subsidized housing, end-of-contract gratuity (25%), return passage and relocation allowance, children's education allowance, medical benefits and car loan.

Details and application form available from **Director of Personnel, National University of Singapore, 10 Kent Ridge Crescent, Singapore 0511.** For information on the Department write to Head, DISCS, NUS, by post or electronically on **ISCHead @ NUS3090**.



Stanford University

Lecturer in Computer Science

Applications are invited for the position of Lecturer or Senior Lecturer of Computer Science beginning in Autumn 1992. Candidates from all areas of computer science will be considered. This two-year appointment is renewable, but not tenure-track.

The Department of Computer Science at Stanford has long been recognized as one of the top computer science departments in the world. For the last six years, the department has offered an undergraduate major which provides excellent computer science education to an exceptionally strong student body. Computer science instruction at Stanford is provided by 28 full-time equivalent faculty members and a staff of four lecturers, who are principally responsible for the introductory core of courses that constitute the computer science major. Although these courses will primarily be at the sophomore/junior level, teaching more advanced courses may also be an option.

Each lecturer is ordinarily expected to teach two courses per quarter. The Department has, however, been very successful in obtaining grant equipment and money for innovative projects and curriculum development work, and lecturers will have the opportunity to participate in such projects to reduce their teaching load to a single course during one quarter of the year. Participation in undergraduate advising and curriculum planning is also expected.

Demonstrated ability and strong interest in teaching undergraduates is essential. An M.S. or Ph.D. in Computer Science is required. Applicants should submit a resume and three references to:

Eric Roberts, Assistant Chair
Department of Computer Science
CS Tresidder, Room A-211
Stanford, CA 94305-3068

Stanford University is an Affirmative Action/Equal Opportunity employer and actively solicits applications from women and minorities.

University of Glasgow

Computing Science Department Lectureship

Applications are invited for a permanent Lectureship in this thriving department of 35 academic staff. We seek an outstanding computer scientist to complement our existing strengths in: human-computer interaction, information retrieval, programming languages, databases, graphics, formal methods and functional programming. You should be able to contribute to the experimental and practical side of large-scale computation though this, by no means, excludes theoreticians. You would join a lively and well-equipped Department, offering opportunities for exciting research and teaching.

Salary will be within the Lecturer scale (£12,860-£23,739 per annum), with placement according to qualifications and experience.

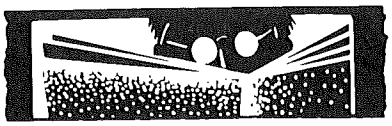
Further particulars may be obtained from the Academic Personnel Office, University of Glasgow, Glasgow, G12 8QQ; or from the Head of the Computing Science Department (tel. 041 330 4463; e-mail to keith@dcs.glasgow.ac.uk).

Applications (8 copies), giving the names and addresses of three references should be lodged on or before 30th June, 1992 with the Academic Personnel Office at the above address. In reply please quote ref. no. 7566.

University of Pennsylvania Library

Director of Library Information Systems

The Director of Library Information Systems provides leadership for info. sys. functions, participates in high level planning, leads in design & execution of academic info. processing systems linked with univ. network & external databases. Works with Assoc. Deans addressing campus computing issues. Reports to the V.P. & Dir. of Lib., manages 9 FTE staff. Manages fin. & elect. resources; designs, supports comp. labs. QUALS: B.A. in field. Adv. degree highly desir. 7 yrs. rel. exp. Knowl. of universities, research libraries, & their info. sys. pref'd. Exp. leading an innovative project highly desirable. Evidence of exc. interpersonal, mgmt. & comm. skills. Teamwork, negotiation skills req'd. Famil. w/open sys. design, standards, client-server apps., microcomp. CD applicns. pref'd. Combined mainframe & microcomp. applications, IBM & UNIX-based sys. pref'd. Fam. w/ library integrated sys. desirable. SAL: Commensurate w/quals. Send a cover letter, resume, and names, add., & phone #s of 3 ref. to Edna Dominguez, Pers. Admin., 3420 Walnut St., Van Pelt Library, Phila., PA 19104-6206. (215) 898-7568. Applications recd. by 4/30/92 will have received first consideration. However, APPLS. WILL BE ACCEPTED UNTIL FINAL CANDIDATE IS SELECTED. AA/EOE Employer



Massachusetts Institute of Technology

Research Scientist
Artificial Intelligence Laboratory

Individual needed to provide system support for research in symbolic and numerical computing. We are seeking an outstanding system designer who will assume primary responsibility for the continued development of MIT's implementation of the Scheme dialect of Lisp, and the associated Unix environment. Other duties include continuing research on high-performance implementations of Lisp-like languages, and supervising students and staff members.

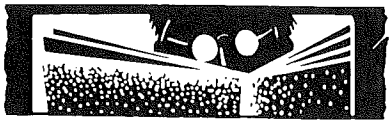
A formal degree (bachelor's level or higher) in Computer Science is preferred. Applicants must have experience with Scheme and with Scheme compilation techniques, and must have developed **substantial** applications in Scheme or Lisp. Applicants must also have extensive knowledge of Unix, although they should have sufficiently good programming taste to not consider this an achievement.

Qualified applicants, please send two copies of both resume and cover letter, referencing **Job No. R92-060** to: **Mr. James McCarthy, MIT Personnel Office, Bldg. E19-239, 77 Mass. Avenue, Cambridge, MA 02139-4307.** MIT is an Equal Opportunity/Affirmative Action Employer. MIT is a non-smoking environment.

RAND
Senior Computer Scientist

RAND is seeking a senior computer scientist for its Santa Monica location to lead research in one of its areas of advanced computing research, which emphasize, but are not necessarily limited to, modeling, simulation and visualization. A successful candidate will have an established research reputation with a demonstrated capacity to lead computing research. A PhD in computing or related field is necessary plus an ability to work collaboratively with researchers in other disciplines. Candidate should also have extensive knowledge of the computing and communications disciplines, and familiarity with the major governmental agencies sponsoring research in these areas. RAND is a private, non-profit research institution engaged in research and analysis of matters affecting national security and the public welfare. U.S. citizenship required. Address applications to: Anthony C. Hearn.

RAND
1700 Main Street
P.O. Box 2138
Santa Monica, CA 09407-2138
An Affirmative Action Employer



The University of Tennessee
Department of Computer Science
Knoxville, Tennessee 37996-1301

The Department of Computer Science seeks to fill one tenure-track faculty position at the rank of Professor, Associate Professor or Assistant Professor, as credentials warrant, beginning Fall 1992. For a full-professorship, a strong research record in the areas of operating systems, scientific computing or software engineering is sought but all major fields in computer science may be considered. Experience directing doctoral students is especially important. Applicants for Associate Professor should have a strong research record, preferably in the above-named areas; experience directing doctoral students is desirable. Applicants for Assistant Professor should have a strong interest in research, preferably in the above-named areas. Applicants for all positions should have a doctoral degree in computer science or a related area. Applicants should specify the rank for which they are applying.

Departmental SUN, IBM and DEC workstations abound for students and faculty and are fully networked. In addition, the department has acquired a Thinking Machine CM-5. The department and the Mathematical Sciences Section of the Oak Ridge National Laboratory jointly operate the Advanced Computing Laboratory which includes full networked Intel iPSC/860, 128 processors; iPSC/2, 64 processors; two Sequent Balances and a Sequent Symmetry; a Stardent Titan with four processors; Cogent; N-Cube; Kendall Square Research machine with 32 processors; and various file servers. Also, Oak Ridge National Laboratory is acquiring an Intel Paragon. In addition, the department has recently received an NSF Small-Scale Infrastructure Award. The department is part of the National Science Foundation Science and Technology center for Research in Parallel Computing. The University operates an IBM 3090 and a large VAX cluster.

Please respond to search@cs.utk.edu. The mailing address is Search Coordinator, Department of Computer Science, 107 Ayres Hall, The University of Tennessee, Knoxville TN 37996-1301.

The University of Tennessee is an EEO/AA/TITLE IX/SECTION 504/ADA employer.

National Oceanic and Atmospheric Administration
Supervisory Computer Specialist

The Space Environment Laboratory (SEL) of the National Oceanic and Atmospheric Administration (NOAA), Boulder, Colorado, is seeking a Supervisory Computer Specialist, GM-334-14 ($54,607–$70,987) to serve as Chief of its System Development Branch. The SEL mission is to acquire and disseminate monitoring information about the solar and near-Earth space environment, prepare and disseminate forecasts of conditions in the space environment, conduct research in solar-terrestrial physics, and develop and apply techniques to improve the monitoring and forecasting of the environment. The System Development Branch consists of approximately 10 computer specialists, mathematicians, physicists, and electronic engineers developing data systems used for all aspects of the laboratory mission. There is an emphasis on real-time operation with data flowing from a variety of satellites, observatories, and sensors in the United States and in other countries. Processing is distributed over several systems, data are exchanged in real time with other federal agencies, and state-of-the-art networks are used. Development of a research data base, development of the next generation of real-time forecast and analysis systems, and research and forecasting use of data streams from new satellites are included in laboratory plans. The individual should have strong management and leadership abilities, be effective in written and oral communication, and have a strong computer background in scientific oriented applications, writing applications programs (e.g., C & Fortran), and advanced methods and techniques in software design (functional, modular, distributed, real-time, network, etc.). The employee will interact with other NOAA laboratories, other federal agencies, and public and commercial organizations.

Copies of the vacancy announcement listing qualifications, selective and quality ranking factors and instructions on how and where to apply may be obtained by calling Jan Presley at (303) 497-6615. For the hearing impaired telephone (303) 497-3014. Please reference the vacancy announcement number:

MASC/NOAA 92-136LR

Procedures will include filing an Application for Federal Employment (SF171, June 1988 edition only) to U.S. Department of Commerce, NOAA/Personnel Division, MC24, 325 Broadway, Boulder, CO 80303-3328. Applications must be received in the Personnel Office by close of business July 2, 1992.

The U.S. Department of Commerce is an equal opportunity employer. U.S. citizenship is required.

ACM POLICY ON NONDISCRIMINATORY ADVERTISING

ACM accepts recruitment advertising under the basic premise that the advertising employer does not discriminate on the basis of age, color, race, religion, gender, sexual preference, or national origin. ACM recognizes, however, that laws on such matters vary from country to country and contain exceptions, inconsistencies or contradictions. This is as true of laws of the United States of America as it is of other countries.

Thus ACM policy requires each advertising employer to state explicitly in the advertisement any employment restrictions that may apply with respect to age, color, race, religion, gender, sexual preference, or national origin. Observance of the legal retirement age in the employer's country is not considered discrimination under this policy.

Peter G. Neumann

LEAP-YEAR PROBLEMS

INSIDE RISKS

To begin 1991, "Inside Risks" considered the effects of computer clock rollovers and other temporal fugits. After Feb. 29, 1992, it seems appropriate to revisit the effects of leap years. The following episodes [1] are worth retelling.

- One of Rob Slade's home computers would not accept a Feb. 29, 1992 date (Saturday), but still managed to know that Mar. 1, 1992 was a Sunday.
- Paul Eggert's contribution for International Software Calendar Bug Day observed that Prime Computer's **MAGSAV** failed at midnight. However, since Prime's 800 number does not work on Saturdays, they probably did not hear about it as much as they would have had it happened on a weekday. G.M. Lack noted that **MAGSAV** probably failed on leap-day because it tried to increment the year by one to set a tape label expiration date, and the resulting nonexistent date Feb 29, 1993 threw it for a loop. Eggert added, "Alas, dates are tricky, and that goes double for date arithmetic."
- Jaap Akkerhuis reported that Imail caused systems to crash worldwide on leap-day, the mail handlers did not recognize the date.
- Roger H. Goun reported that the PC MS/DOS mail system *UUPC/extended* hung the PC on leap-day. "Drew Derbyshire, the author of UUPC, traced the problem to a bug in the mktime() library function in Borland C++ 2.0, which converts a time to calendar format. Drew demonstrated that mktime() will hang a PC on Leap Day, and reported the problem to Borland. As distributed, UUPC is compiled with Borland C++ 2.0, though source code is available for do-it-yourselfers . . . Drew tried to warn UUPC users by mail after discovering the problem on Saturday. Ironically, many did not get the message until Sunday or Monday, when they found their PCs hung in UUPOLL."
- Rhys Weatherley noted that a Windows 3.0 newsreader using Borland C++ 2.0 locked up, due to a bug in 'mktime' converting to Unix data/time formats, although the problem may have been due to the run-time library.
- Mark Brader raised some questions about digital watches going from Feb. 28 to Mar. 1. Several people responded that their watches always assumed 31 days, and a manual adjustment was required otherwise.
- Douglas W. Jones noted that all liquor licenses in Iowa expired on the 28th, and the new licenses were not in force until the 1st. The state announced that this was due to a 'computer error' and promised not to enforce the law on leap-day for establishments caught in the glitch.

The conclusions have not changed much since our Jan. 1991 column [1]. You might think that the leap-day problems would have been adequately anticipated, especially in the four years since their previous incarnation, as noted in [1]. Getting clock arithmetic correct might seem to be a conceptually simple task—which is why it is not taken seriously enough. But even if earlier leap-year problems were caught in older systems, they continue to recur in newer systems. So, now we have four more years to develop new software with old leap-year bugs, and perhaps even some creative new ones!

Distributed System Woes

Leslie Lamport once said, "A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable." As we evolve more toward distributed systems, there is a pervasive failure to realize that certain design decisions must change, particularly that we must no longer rely on weak-link centralized servers.

After several painful experiences, I am compelled to issue a cry to the system developers that, in our modern times, distributed systems should not have so many weak links. There are typically all sorts of hidden dependencies—on password servers, file servers, name servers, system logs, and lock managers—whose malfunctions can cause pervasive disturbances. Caveh Jalali noted how timeouts are nested, assuring total blockage under certain circumstances. In the research community and in the Tandem world, we know how to drastically reduce the dependence on weak links. It is time that developers of distributed operating systems got the message: distributed systems are a step forward only if they are not disabled by simple server failures.

Some of you will say you do not have this problem because you have a stand-alone personal computer or totally centralized system. Congratulations for successfully living in the past. That works just fine as long as *your* system is up! The corresponding situation in distributed systems is that things work wonderfully as long as every weak link you depend on is OK. But the likelihood of success is even lower in distributed systems unless the weak links have been avoided by good system design. The problems of designing robust distributed systems present a major challenge.

Conclusions

There is a serious lack of system sense and consistent software practice. Lessons are ignored, and heads remain stuck in the sand, ostrichlike. It is time to reform.

1. Neumann, P.G. ACM SIGSOFT *Softw. Eng. Notes 17*, 2 (Apr. 1992).

SPRINGER FOR COMPUTER GRAPHICS

N. MAGNENAT THALMAN and D. THALMAN

CREATING AND ANIMATING THE VIRTUAL WORLD

A collection of papers, from the fourth International Workshop on Computer Animation held in Geneva 1992, containing original research results and techniques in various areas of computer animation. Topics selected include physics-based animation, human animation, and geometric modelling and animation.

1992/APPROX. 250 PP., 131 ILLUS./HARDCOVER
$120.00 (TENT.)/ISBN 0-387-70093-5
COMPUTER GRAPHICS SOCIETY

G. GARCIA and I. HERMAN

ADVANCES IN COMPUTER GRAPHICS VI

Images: Synthesis, Analysis, and Interaction

Offers a range of tutorials on image synthesis and image reconstruction, based on the Eurographic '90. The unique combination of the two fields of interest is not only constructive but necessary to a whole range of theoretical problems and applications. Contains high level tutorials addressing visualization, image processing, X/PEX programming, intelligent CAD systems, image reconstruction, and human visual systems.

1991/460 PP., 86 ILLUS./HARDCOVER $117.00
ISBN 0-387-53455-5
EUROGRAPHIC SEMINARS

R. LATHAM

THE DICTIONARY OF COMPUTER GRAPHICS TECHNOLOGY AND APPLICATIONS

The first dictionary covering software, hardware, and applications in computer graphics, this book offers definitions of terms not found anywhere else. The author guides both novices and specialists alike through the maze of terminology surrounding one of the most exciting growth areas in computing. An excellent reference for anyone who would like to become familiarized with technical jargon in applications and allied technologies.

1991/160 PP., 19 ILLUS./HARDCOVER $24.95
ISBN 0-387-97540-3

W.A. GAMAN and W.A. GIOVINAZZO

PHIGS BY EXAMPLE

Following the principle that the best way to learn PHIGS is by using PHIGS, this book takes an integrated approach to learning the program. Written for the novice programmer, the authors begin by exploring the basic concepts of PHIGS and ends with more advanced topics of PHIGS+. Areas of interest include creating, shading, coloring, and interactively manipulating geometric objects.

1991/218 PP., 26 ILLUS./HARDCOVER $29.95
ISBN 0-387-97555-1

J.L. ENCARNAÇAO, R. LINDNER, and E.G. SCHLECHTENDAHL

COMPUTER AIDED DESIGN

Fundamentals and System Architectures
2nd edition

Describes the principles, methods, and tools that are commonly used in the application of computers to design tasks. This new edition is completely revised and updated with extensive new materials - including computer graphics, implementation methodology, and CAD data transfer.

1991/423 PP., 240 ILLUS./HARDCOVER $69.00
ISBN 0-387-52047-3
SYMBOLIC COMPUTATION

N. MAGNENAT THALMANN and D. THALMANN

COMPUTER ANIMATION

Theory and Practice
2nd revised edition

Now in its second edition, **Computer Animation: Theory and Practice** continues to stand out as an excellent presentation of all aspects of computer animation. Presents computer-assisted animation techniques such as key-frame interpolation and use of paint and color, as well as detailed information on the state-of-the-art in computer animation and a history of animation systems and languages.

1991/245 PP., 156 ILLUS./HARDCOVER $49.00
ISBN 0-387-70051-X
COMPUTER SCIENCE WORKBENCH

D.F. ROGERS and R.A. EARNSHAW

STATE OF THE ART IN COMPUTER GRAPHICS

Visualization and Modeling

This third volume, from the State-of-the-Art in Computer Graphics Summer Institute, contains new and original work at the cutting edge of computer graphics. Topics include: parallelism, radiosity, modelling, geometry of graphics, and user interfaces.

1991/368 PP., 171 ILLUS./HARDCOVER $79.00
ISBN 0-387-97560-8

P. WISSKIRCHEN

OBJECT ORIENTED GRAPHICS

From GKS and PHIGS to Object-Oriented Systems

This book covers computer graphics programming using object-oriented philosophy and its programming paradigms. The author shows how the use of object-oriented techniques can lead to a more powerful and flexible graphics system than that of GKS and PHIGS. An object-oriented multi-level system GEO++ is defined as a basis for detailed examples using Smalltalk-80. Readers will find this to be an invaluable text for bridging the gap between traditional graphics programming systems and the object-oriented approach.

1990/236 PP., 83 ILLUS./HARDCOVER $39.00
ISBN 0-387-52859-8
SYMBOLIC COMPUTATION

R. RONCARELLI

THE COMPUTER ANIMATION DICTIONARY

" This book will be useful for the computer hobbyist, the student, and the graphics professional. Being succinct and exceptionally clear in its definitions, this book is fast and easy to use." **- Preview, 1990**

1989/124 PP./SOFTCOVER $28.00
ISBN 0-387-97022-3

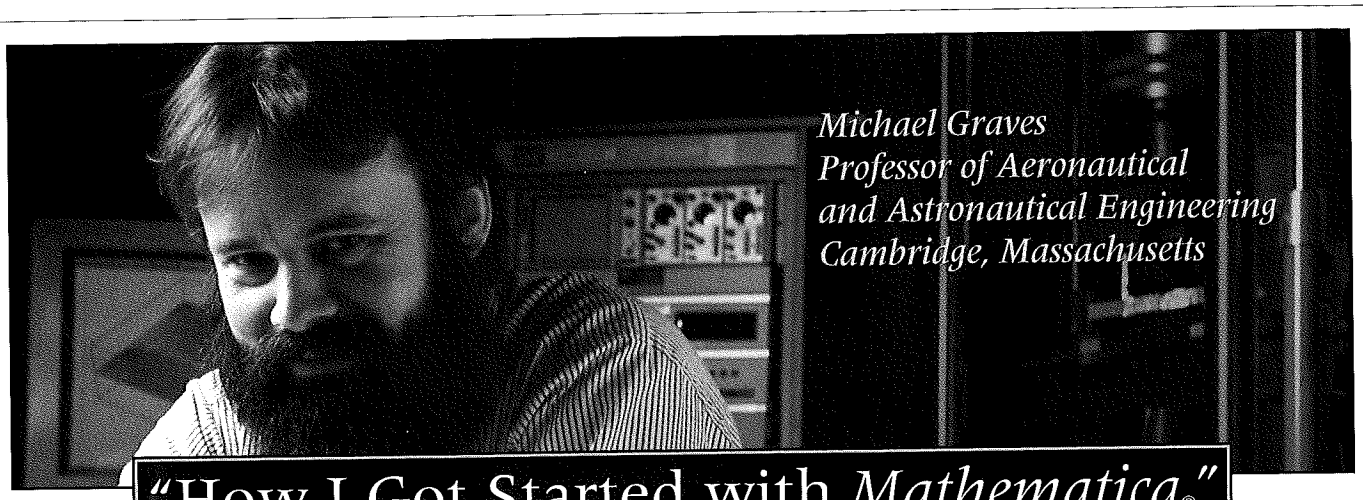
SPRINGER 150 1842-1992 FOR SCIENCE

To Order:

Call Toll Free 1-800-SPRINGER (in NJ call 201-348-4033) or visit your local technical bookstore today!

5/92 REFERENCE #: S905

Circle #11 on Reader Service Card



Michael Graves
Professor of Aeronautical and Astronautical Engineering
Cambridge, Massachusetts

"How I Got Started with *Mathematica*®"

A few years ago, I was working with another engineer at Boeing on a project for the Air Force that required extensive symbolic manipulations. In particular, we were looking for series solutions to partial differential equations. My colleague said, "Why don't we try *Mathematica*? It does exactly what we want to do." But I'd never used it.

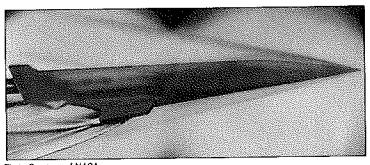


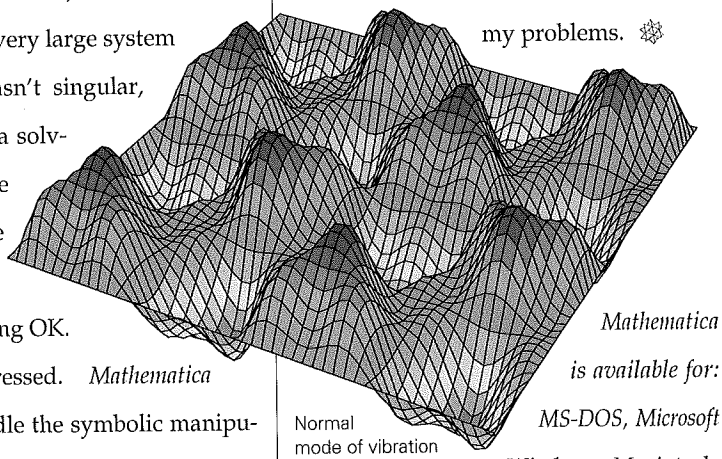
Photo Courtesy of NASA

We had programmed the equations in Fortran and were getting numerical instability errors. We knew that was a mistake—the equations were well behaved, the matrices were nonsingular. But because of the limits of the significant digits we got with Fortran, we were running into this problem. Although I hadn't used it, I was willing to try sorting out the problem with *Mathematica*.

Using *Mathematica*, we discovered that indeed this very large system of equations wasn't singular, that it did have a solvable set. So we rewrote our code knowing everything was working OK.

I was impressed. *Mathematica* was able to handle the symbolic manipulations we needed and more. It solved differential equations and gave exact solutions. It made plots for me on the screen. It gave me output that I could read and understand, and the input was very natural. I'm used to Macintosh systems, but even on a workstation I could work with *Mathematica* in a more or less intuitive way.

Now I turn to *Mathematica* whenever I've got any questions about symbolic math. I don't worry about the limits of significant digits in Fortran, C, or any other programming language because with *Mathematica* I can maintain hundreds of significant digits with no concerns about inaccuracies or errors. *Mathematica* was the answer to my problems. ❁



Normal mode of vibration of a clamped plate.

Mathematica is available for: MS-DOS, Microsoft Windows, Macintosh, CONVEX, DG AViiON, DEC VAX (ULTRIX and VMS), DEC RISC, HP 9000, HP Apollo, IBM RISC System/6000, MIPS, NeXT, Silicon Graphics, Sony, Sun-3, and SPARCstations.

Prices in U.S. and Canada start at $595.

Educational discounts are available.

Orders: 1-800-441-MATH

Mathematica® 2.0
The Standard for Technical Computing

Wolfram Research, Inc., 100 Trade Center Drive, Champaign, IL 61820-7237, USA; 1 (217) 398 0700; fax: 1 (217) 398 0747; email: info@wri.com
Wolfram Research (UK) Ltd., P.O. Box 114, Abingdon, Oxon OX13 6TG, United Kingdom; +44 (235) 550 440; fax: +44 (235) 550 445; email: info-uk@wri.com